SailPoint™

MANAGING THE BUSINESS OF IDENTITY

SailPoint IdentityIQ

Version 6.1-1

Integration Guide

Table of Contents

Chapter 1: Overview

The following topics are discussed in this chapter:

What is SailPoint IdentityIQ?

SailPoint IdentityIQ is unique and comprehensive Identity Governance solution that addresses the digital security concerns of today's global enterprises. The product provides automated analysis, monitoring and management of risk associated with access to critical or sensitive applications and data via centralized visibility to and control of identity data across the enterprise. Through the use of patent-pending technologies and annalistic, IdentityIQ improves security, lowers the cost of operations, and improves an organization's ability to meet compliance and governance demands.

SailPoint IdentityIQ is a business-oriented identity governance solution that delivers risk-aware compliance management, adaptive role management, access request management, and identity intelligence. Some of the world's largest organizations are using IdentityIQ to improve security, minimize risk and streamline their compliance efforts.

IdentityIQ minimizes the access burden placed on IT staff by empowering end users across the organization to request and manage their own access.

Organizations strive for better visibility into potential risk factors across their business. With Identity Intelligence from IdentityIQ, organizations can transform technical identity data scattered across multiple enterprise systems into centralized, easily understood and business-relevant information. The visibility and insights offered by IdentityIQ through dashboards, risk metrics and reporting provide a clear understanding of identity and access information and help to proactively manage and focus compliance efforts strategically across even the most complex enterprise environments.

SailPoint IdentityIQ components

SailPoint IdentityIQ includes the core components illustrated in Figure 1—SailPoint IdentityIQ components.

SailPoint IdentityIQ components

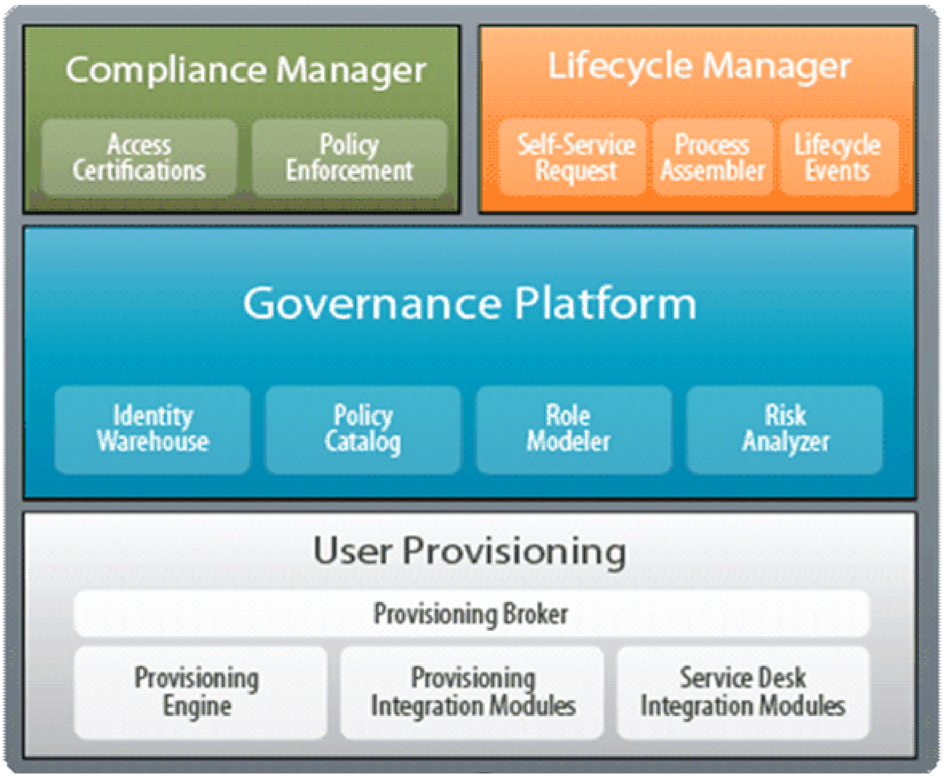


Figure 1—SailPoint IdentityIQ components

Together, these components provide a unique and comprehensive identity governance solution. The components are:

- **Compliance Manager**: streamlines complex compliance processes for greater efficiency, effectiveness and lower costs. Its integrated risk model provides the contextual framework that enables organizations to prioritize compliance activities and focus controls on the users, resources, and access privileges that represent the greatest potential risk to the business
- **Lifecycle Manager**: is a highly flexible solution for managing changes to user access and automating the initiation of provisioning activities within an enterprise. It maps directly to the lifecycle of a user within an organization (joining / moving / leaving) and the core identity business processes which support these lifecycle events (provision / change / de provision)
- **Governance Platform**: to enhance compliance performance, improve security and reduce risk, Lifecycle Manager leverages the IdentityIQ Governance Platform to align provisioning requests with an organization's pre-established identity governance model. SailPoint's unique combination of roles, policy, and risk provides a strong framework for evaluating all requests for changes to access against predefined business policies.
- **Resource Connectivity**: The resource connectivity consists of the following modules:
    - IdentityIQ Provisioning Engine
    - Provisioning Integration Module
    - Service Desk Integration Module

This document provides a guide to the integration between the following products and IdentityIQ:

- IdentityIQ Provisioning Engine
- Provisioning Integration Modules
    - SailPoint IdentityIQ BMC Identity Management Suite Platform Integration Module (IdM Suite PIM)
    - SailPoint IdentityIQ Oracle Identity Manager
    - SailPoint IdentityIQ Sun Identity Manager Provisioning Integration Module
    - SailPoint IdentityIQ Novell Identity Manager
    - SailPoint IdentityIQ IBM Tivoli
- Service Desk Integration Modules
    - SailPoint IdentityIQ Remedy Ticketing SIM
    - SailPoint IdentityIQ Integration for ServiceNow

This document is intended for the above products and IdentityIQ System Administrators and assumes a high degree of technical knowledge.

Section 1: IdentityIQ Provisioning Engine

This section provides the necessary installation and configuration steps with which to configure connectors (Connector Manager and Connector) as an individual application within IdentityIQ for compliance and provisioning activities.

This section contains the information on the following:

- IdentityIQ Provisioning Engine on page 13

Chapter 2: IdentityIQ Provisioning Engine

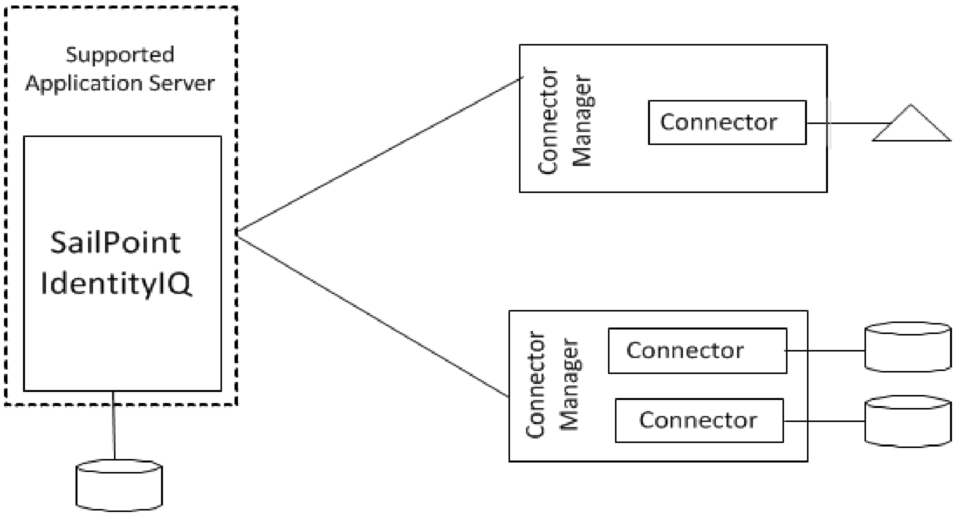
The following topics are discussed in this chapter:

Overview

The SailPoint IdentityIQ Provisioning Engine allows IdentityIQ users to interact and perform the following requests on the supported applications:

- Aggregation
- Provisioning Entitlements and Role Assignments
- Password Management

The Connector Manager and Connector as shown below execute the above requests sent by the Provisioning Engine on the target application:



The Connector Manager and the companion Connectors provide communications between User Administration components and the managed system. Connector Manager encrypts, queues and transmits administration requests to the connector, which executes it on the managed system. Connector Manager has now been enhanced so that multiple IdentityIQs can directly connect to it without the connector gateway component.

The Connectors are bound to the application they manage. Each supported application has an associated Connector. To service requests for a given application, it's associated Connector is required. For instance, to perform provisioning requests on Active Directory the Connector for Active Directory is used. Refer the "Applications supported and associated connectors" on page 17 for the complete list of the supported applications and their associated Connector.

The Connector Manager has the ability to host one or more Connectors of the same or different type with the exception of the Active Directory Connector, which requires a dedicated Connector Manager.

The information exchanged between IdentityIQ and the Connector can be secured using an encrypted channel.

This document provides the necessary installation and configuration steps with which to configure connectors (Connector Manager and Connector) as an individual application within Identity IQ for compliance and provisioning activities.

For some of the supported applications like RACF and AS 400, the components corresponding to the Connector Manager and Connector are refered to as Agents. In this document, anything regarding Connector Managers and Connectors, applies to Agents unless specified.

Note: CONTROL-SA/Agent requires Connector Gateway.

Supported features

SailPoint IdentityIQ Provisioning Engine supports the following features:

- **Password Interception**: SailPoint IdentityIQ provides the capability of Password Interception, by which password of an account changed on managed system is intercepted to IdentityIQ.

  New password is intercepted amd sent to IdentityIQ to be propogated to other managed systems on which the account is defined.

  For proper functioning of the Password Interception, the Password Interception settings are mandatory. For more information on setting the Password Interception from IdentityIQ, see "Settings for configuring Password Interceptor" on page 23.
- **Group provisioning**: Group provisioning is supported for all the connectors in IdentityIQ. Group provisioning supports add, delete and update group.

  Perform one of the following based on the existing or newly created applications:

  - (*For newly created applications*): Enable the account group management flag from **LCM=>Additional options**.
  - (*For existing applications*) Perform the following:
    a. Enter **GROUP_PROVISIONING** in feature string of application.
    b. Enable account group management flag from **LCM=>Additional options.**
    c. Add group provisioning policy for create group and update group.
    d. Required keyword for group provisioing needs to be added in the schema.
- **Current Password field for self service**: To enforce user to enter current password while changing password through self-service mode, **CURRENT_PASSWORD** string must be added in the feature_string of the respective application.

Configuration

Configuring the Provisioning Engine involves defining an Application in IdentityIQ for each of the target applications. When creating an Application within IdentityIQ, the following information is required:

Application type

- Select the type of application that IdentityIQ is managing.
- See the "Applications supported and associated connectors" on page 17 for the list of supported applications and their respective application types.

Connector Gateway details

Each application IdentityIQ is managing must have a dedicated Connector Gateway. The following are details of the Connector Gateway required when defining an Application:

- Connector Gateway host: Hostname/IP of the machine where the Connector Gateway for the targeted Application is installed.
- Connector Gateway port: the port on which the Connector Gateway is listening.

Note: Connector Manager version 6.0 and above does not require Connector Gateway. Connector Manager's hostname and port number can be provided.
CONTROL-SA/Agent still requires Connector Gateway.

Connector details

A single Connector manages one or more instances of an associated application type. Each instance of the application managed by a Connector requires a separate configuration to create a configuration set unique to that instance of the application. The information below is required to establish this differentiation:

Note: The values of the attributes referenced below are case sensitive and if provided incorrectly, will cause requests from IdentityIQ to fail.

- MSCS Name: the name for the managed system configuration set that is defined when installing the Connector Manager and Connector.
- Username: user account used when authenticating to the Target Application through the Connector.
- Password: password for the user account specified in Username.
- Container: if the application supports container, a semicolon separated list of containers to search for account and group objects.

Encryption details

The communication between IdentityIQ and the Connector can be secured by using either DES or Triple DES encryption algorithm. For more information on enabling secured communication, see "Enabling secured communication between IdentityIQ and Connector Manager" on page 20. The following information is required to enable secured communication:

- **Encryption**: the Encryption Algorithm used to secure the communication channel.

- **Encryption Key File**: fully qualified file name of the encryption key file on the IdentityIQ server. For more information on enabling secured communication, see "Enabling secured communication between IdentityIQ and Connector Manager" on page 20.

Click Test Connection to check if entered values are correct.

Note: When aggregation is in progress, do not perform the test connection.

Synchronizing IdentityIQ and Connector Manager

Setting the value of **syncSchema** in the IdentityIQ SystemConfiguration object to **true** synchronizes all application schema attributes of an application created in IdentityIQ with the MSKWDS.PRM file of Connector Manager. Connector Manager uses this file for all of its transactions. The default value of syncSchema is **true**.

If the Connector Manager instance is also part of a BMC Identity Manager deployment, IdentityIQ should be configured to not synchronize the application schema since the MSKWDS.PRM file will be created when requests are sent from BMC Identity Management. In this circumstance, set the value of **syncSchema** to **false** to avoid schema synchronization between IdentityIQ and Connector Manager.

The **syncSchema** attribute is also configurable at the application level within IdentityIQ. This allows the enabling or disabling of application schema synchronization when a Connector Manager instance is part of a BMC Identity Management deployment and others are not.

Testing the Application

The Test Connection button in the Application definition web page can be used to test basic connectivity between IdentityIQ and the managed application. Additional tests can be run using the IdentityIQ integration console.

1. Consistent with the BMC Identity Management integration guide, launch the console by using the IdentityIQ script in the **WEB-INF/bin** directory of the IdentityIQ installation to run IdentityIQ **integration**.
2. When the command prompt appears enter the following:
   ```
   list
   ```
   This lists the names of all Applications created in system.
3. From the console enter the following:
   use "SM AD Application"

   This makes the Application active for further console commands.
4. Enter the following:
   ping

   The following response is displayed:

   Response: Connection test successful

   If you do not see this response, check the Application Server logs to verify the request is being sent to Connector Manager. Enable **log4j** tracing on this class within the Application Server using the following command:

   log4j.logger.sailpoint.connector.sm=debug

   This lets you verify the requests are transmitting over the network, and how they are being processed.

Applications supported and associated connectors

The following table lists the supported applications and associated connectors:

Table 1—Supported applications and associated connectors

Application	Connector to be used	Application Type	MSCS Type
RACF	CONTROL-SA/Agent for RACF	RACF - Full	RACF
CA-ACF2	CONTROL-SA/Agent for CA-ACF2	ACF2 - Full	ACF2
CA-TopSecret	CONTROL-SA/Agent for CA-Top Secret	TopSecret - Full	TSS
AS-400	CONTROL-SA/Agent for AS/400	AS400 - Full	AS400
IBM DB2	CONTROL-SA/Agent for DB2 Universal Database	DB2 - Full	DB2

Target Aggregation implementation in Provisioning Engine for Mainframe Connectors

The “Target Aggregation” functionality is supported only for the following Provisioning Engine based Mainframe Connectors:

- Top Secret
- CA-ACF2
- RACF

Implementing Target Aggregation

Execution of Target Aggregation task triggers the aggregation and IdentityIQ sends the message to fetch the resources and permissions on the resources in the following sequence:

- 1 - Synchronize the keywords
- 2 - Load resource permissions
- 3 - Update IdentityIQ database
- 4 - Run certification process to revoke permission

1 - Synchronize the keywords

To get the resources send the following keywords for resource and ACE entity:

- **Structure fields for Resource**:

  RES_ID

  RES_TYP

  RES_OE_PR
- **Structure fields for ACE**:

  ACE_TYPE

  ACE_UG

ACE_USR

ACE_OE

ACE_ATTR

ACE_PLCE

The predefined fields are mentioned in the **connectorRegistory.xml** file with the following respective key values:

- (For Resource) **splResourceAttributes**
- (For ACE) **splAceAttributes**

For example, For Top Secret managed system

```
<entry key="splResourceAttributes">
  <value>
    <Map>
      <entry key="RES_HAS_OWNER" value="false" />
      <entry key="RES_OWNER" value="false" />
    </Map>
  </value>
</entry>
<entry key="splAceAttributes">
  <value>
    <Map>
      <entry key="FACILITY" value="true" />
      <entry key="EXPIRES" value="false" />
      <entry key="ACCESS" value="true" />
      <entry key="DAYS" value="true" />
      <entry key="TIMES" value="false" />
      <entry key="LIBRARY" value="false" />
      <entry key="PRIVPGM" value="true" />
      <entry key="ACTION" value="true" />
      <entry key="VMUSER" value="true" />
      <entry key="APPLDATA" value="false" />
      <entry key="SCRIPTNAME" value="false" />
      <entry key="SCRIPTPARM" value="false" />
    </Map>
  </value>
</entry>
<entry key="splTargetPermissionsInterestingKwds">
  <value>
    <Map>
      <entry key="FACILITY" value="true" />
      <entry key="EXPIRES" value="false" />
      <entry key="ACCESS" value="true" />
      <entry key="DAYS" value="true" />
      <entry key="TIMES" value="false" />
      <entry key="LIBRARY" value="false" />
      <entry key="PRIVPGM" value="true" />
      <entry key="ACTION" value="true" />
      <entry key="VMUSER" value="true" />
    </Map>
  </value>
</entry>
```

The keywords mentioned in **splTargetPermissionsInterestingKwds** key are set of keywords that will be used to form the permissions string. These keywords are not sent to Mainframe Connector while keyword synchronization.

If any keyword needs to be added in the permissions list then **splTargetPermissionsInterestingKwds** tag and perform target aggregation to populate value of the keyword on record.

2 - Load resource permissions

1. Configure unstructured targets for the application you want to execute target aggregation.
   Provide the following details on **Unstructured Target Configuration:**
   - **Name**: Provide name to the created unstructured configuration.
     For example, **RACFUnstructuredTargetConfig**.
   - **Correlation Rule**: Create a correlation rule to correlate the permissions and entities in IdentityIQ.

   The following sample code can be used for correlation:

```
import sailpoint.api.Correlator;
import sailpoint.tools.xml.XMLObjectFactory;
      private String ATTR_USER_OBJECT= "USER_ID";
      private String ATTR_GROUP_OBJECT= "GROUP_ID";
      Map returnMap = new HashMap();
  if ( isGroup ) {

returnMap.put(Correlator.RULE_RETURN_GROUP_ATTRIBUTE,"nativeIdentity");
          returnMap.put(Correlator.RULE_RETURN_GROUP_ATTRIBUTE_VALUE,
nativeId);
      } else {
      returnMap.put(Correlator.RULE_RETURN_LINK_IDENTITY, nativeId);
}
    System.out.println("Incomming ["+nativeId+"] correlated
["+XMLObjectFactory.getInstance().toXml(returnMap)+"]");
      return returnMap;
```

2. Depending on the type of the mainframe connector, select **Target Source type** and other attributes:
   - For application type **RACF Full**:

     Target Source type: **PE2 RACFCollector**

Attributes	Description
Name of RACF Application	Provide the name of the application.
Name of RACF Target	Provide the target that needs to be aggregated. * to aggregate for all targets.
Target Resource Type	Provide the type of target that needs to be aggregated.
Unit	
Volume	
RACF Type	The value can be ID or PREFIX.
Generic Type	The value can be Yes or No.

- For application type **Top Secret Full**:

  Target Source type: **PE2 TSSCollector**

Attributes	Description
Name of TSS Application	Provide the name of the application.
Name of TSS Target	Provide the target that needs to be aggregated. * to aggregate for all targets.
Target Resource Type	Provide the type of target that needs to be aggregated.
TSS Type	The value can be ID or PREFIX.

- For application type **CA-ACF2**:

  Target Source type: **PE2 ACF2Collector**

Attributes	Description
Name of ACF2 Application	Provide the name of the application.
Name of ACF2 Target	Provide the target that needs to be aggregated. * to aggregate for all targets.
Target Resource Type	Provide the type of target that needs to be aggregated.
ACF2 Type	The value can be NEXTKEYS or PREFIX.

When the task of target permissions is executed, Provisioning Engine should send connector a request to load resource permissions in IdentityIQ.

The resource name needs to be provided in **Unstructured Target Configuration** which can be fetched from Managed System.

3 - Update IdentityIQ database

The resources and permissions will be updated in the respective account as a string in delimited format. Customer can define the delimiters for field separator and list separator by modifying the following keys:

```
<entry key="fieldDelimiter" value="#"/>

<entry key="listDelimiter" value=";"/>
```

The default values for field and list delimiter are **#** and **;** respectively.

4 - Run certification process to revoke permission

To revoke permission through certification run **Refresh Identity Cube** and **Refresh Entitlement Correlation** tasks before running certification and follow the certification process.

Enabling secured communication between IdentityIQ and Connector Manager

This section provides the steps used to enable secured communication between IdentityIQ and Connector Manager.To secure the communication channel between IdentityIQ and Connector Manager, IdentityIQ uses either DES or Triple DES for encrypting the exchanged data. Regardless of the type of encryption used, an encryption key file is required. You can use one common encryption key file for all the Connector Managers or use separate key files for each Connector Manager.

On IdentityIQ computer

To create an encryption key file, perform the following steps for each key file required:

1. From the command prompt change the directory to ***IIQFolder*\WEB-INF\bin\<*OS_Flavor*>**
2. Run the following command:
   kgen <encryption_key_file_name>

   Note: The complete *fileName* (including the path and extension) is asked while creating an IdentityIQ Application.

On Connector Manager/Agent computer

Perform the following steps for each Connector Manager/Agent.

Connector Manager

1. Copy the encryption key file created above to the ***cmHome*\conf** folder.
2. Run the Connector Manager Administration Console.
3. Select the Connector Manager instance and click Connector Parameters.
4. Modify the parameters as shown below:
   FOREVERY ENCR_EXT_ACT: set the value to **Y**

   FOREVERY FILE_ENCREXT: set the value to **%CTSROOT%\CONF\*keyFileName***

Agents

1. Copy the encryption key file created above to <AGENT_INSTALL>\DATA folder.
2. Edit file <AGENT_INSTALL>\DATA\CTSPARM.PRM and modify the parameters as shown below
   FOREVERY ENCR_EXT_ACT: set the value to **Y**

   FOREVERY FILE_ENCREXT: set the value to **%CTSROOT%\DATA\*keyFileName***

Installing and configuring the Connector Gateway

Note: The Java Development Kit 1.5 or later is required.

To achieve optimum performance, the Connector Gateway should be installed on the same computer as the Connector Manager, but it can be installed on any computer with network connectivity to both IdentityIQ and the Connector Gateway.

Create an installation directory on the computer where the Connector Gateway will run and extract the contents of the **ConnectorGateway-<version>.zip** file that is included with IdentityIQ. Use the following OS-specific installation instructions located in the associated README file included in the **ConnectorGateway-<version>.zip**.

Before you begin

- Verify the Connector Manager is not connected to any application.
- The Connector Manager and Connector Gateway can be started / stopped in any order.
- A directory named **log** should always be present in Connector Gateway installation directory. Check the logs created in **log** directory to follow activities occurring in the Connector Gateway. If any problem arises and logs are not generated, check Windows Event viewer.
- JavaService creates a log file, **javaservice.log**. This file may be located in the root directory of the installation partition. This file can contain information about the installation and uninstallation of the Windows service as well as startup and shutdown activities of the service.
- If there is a firewall between Connector Gateway and Connector Manager, then the port on which Connector Manager is listening must be opened.

Additional configuration parameters

Use the following parameters defined in the IdentityIQ SystemConfiguration object and apply to all applications:

- **syncSchema**: If set to **true**, IdentityIQ application schema will be synchronized with the Connector Manager. Default value is **true**.
- **smReadTimeout**: Timeout value in minutes for read operations between IdentityIQ and the Connector Manager. Default value is 10 minutes.
- **multiColumnSeperator**: The delimiter for multi-column attributes in Connector Manager. The default value is '#'. If the default value does not work for an application, add a new separator into the application template as shown in the following example:
  ```
  <entry key="multiColumnSeperator" value="+" />
  ```
- **IBMcharacterSet**: The related character set in java for codepage used on agent side. Default value: IBM500. For more information, see the following link:
  http://www-01.ibm.com/software/globalization/ccsid/ccsid_registered.html

UTF-8 support

For applications supporting UTF-8 data, it is required to add a new attribute (printed below) into the application template.

<entry key="characterSet" value="UTF-8"/>

Note: For an application, whenever the "characterSet" attribute is added or updated, the respective Connector Manager must be restarted for the changes to take effect.

Delta Aggregation

Note: SailPoint IdentityIQ Technologies recommends using the optimized re aggregation option of Account Aggregation instead of using the delta aggregation for the Provisioning Engine based connectors. This inference is based on the results of the performance testing carried out to compare the times taken by both these mechanisms. For using the optimized reaggregation feature, see the Account Aggregation section of the IdentityIQ User guide.

Settings for configuring Password Interceptor

Password Interception is not enabled by default. There are few steps that need to be performed in order to enable it both on the Connector Manager and IdentityIQ.

Note: (For Connector Manager version 5.5 and Agents) Connector Gateway version 6.0 is a prerequisite for Password Interception.
(For Connector Manager version 6.0) Connector Gateway is not required for Password Interception.

To Enable Password Interception on IdentityIQ, add the following attributes in Service definition:

1. **SMListener**: Add the attribute and hostname as follows:

- Add the following attribute in SMListener:

```
<Attributes>
<Map>
    <entry key="applications" value="Application name"/>
    </Map>
</Attributes>
```

Note: To configure multiple applications for password interceptor, add multiple names in application value as follows:
<entry key="applications" value="App1,App2"/>

- Add the hostname as follows:

```
<ServiceDefinition created="1347280738108" hosts="hostName"
id="2c9095d439b02f0b0139b030973c00e9" modified="1347285339111"
name="SMListener">
```

In the preeceding command line, *hostName* is the name of the host where IdentityIQ is installed.

2. **ResourceEvent:** Add the hostname as follows:

```
<ServiceDefinition created="1347280738124" hosts="hostName"
id="2c9095d439b02f0b0139b030974c00ea" modified="1347285354945"
name="ResourceEvent">
```

In the preeceding command line, *hostName* is the name of the host where IdentityIQ is installed.

Note: If the application that is configured to receive password changes was before upgrading IdentityIQ to version 6.0 then perform the following additional steps:

a. Navigate to the application definition page (application that is configured to recieve delta changes).
b. Re-enter the password.
c. Save the application.
d. Perform a **Test Connection**.

When IdentityIQ starts it will start a thread per application for which password interception is enabled. This thread is responsible for receiving the delta changes from that application. If any change is made to the above attributes or if you enable or disable encryption for the application that is configured to receive password changes, the IdentityIQ server must be restarted for the changes to take effect.

Password Interceptor is supported for the following CTSA based connectors:

- CA-ACF2
- CA-Top Secret (TSS)
- AS-400

  To enable Password Interceptor on Connectors, refer to the respective *Connector Administration Guide*.
- RACF

  To enable Password Interceptor on RACF, refer to the SailPoint IdentityIQ *Connector-Agent for RACF Administration Guide*.

Known/Open issues

1. In Connector Manager, when interceptions are enabled, a parameter can be added into the **INTERCEPT_SEND_MAX** SMPARM file. This indicates the maximum number of interceptions that the Connector Manager process will send before pausing to receive acknowledgement for earlier interceptions. The default value of this parameter is "5". This means if the Connector Manager does not get acknowledgement for 5 interceptions, it will stop sending further interceptions.

   When IdentityIQ receives interceptions, it will send acknowledgement of the interception as soon as it receives it. If the IdentityIQ application crashes or application server process is shut down at the time when application was about to send acknowledgement of interceptions, Connector Manager will not receive it and that interception will remain in **pending for acknowledgement** list of Connector Manager. When a similar instance happens 5 times, Connector Manager will stop sending further interceptions.

   Workaround: Restart the Connector Manager to clear the **pending for acknowledgement** list and start sending interceptions again.
2. Connector Manager not sending interceptions. Following message appears in the SOFFI_MSG_.log file:
```
2011/10/28 12:59:30 CTS1182E Queue file is full: C:\Program Files\BMC
Software\CONTROL-SA\Services Manager\AD\WORK\QUEUE.DAT
2011/10/28 12:59:30 CTS1180E <Put> failed for Queue file:
<C:\Program Files\BMC Software\CONTROL-SA\Services
Manager\AD\WORK\QUEUE.DAT>
2011/10/28 12:59:30 CTS1308E <rss_queue_put> Function failed
2011/10/28 12:59:30 CTS1308E <CTSUserEventEx> Function failed
2011/10/28 12:59:30 CTS2310W warning: queue is full. Retrying in 10
seconds.
```

   Workaround: From bin directory, run the **ctsqcr** command which will clear the queue.
3. During aggregation, the following error message appears if the container name is provided incorrectly while creating the application.
```
Exception during aggregation. Reason: java.lang.RuntimeException:
java.lang.RuntimeException: Error while iterating. SailPoint
IdentityIQ.connector.sm.SMException:
D774190558000004OGS01SAILPOINL-EE001026200000000000000411110816464424
2011/11/08 16:46:44 CTS1460I R Download Managed System ldapip started
2011/11/08 16:46:44 CTS1469I R List of containers: * 2011/11/08 16:46:44
CTS1318I R DOWNLOAD Processed.... 0 containers, 0 accounts, 0 groups, 0
```

```
connections 2011/11/08 16:46:44 CTS1308E T <CS_dwnld_oe_subtree> Function
failed 2011/11/08 16:46:44 CTS1308E T <CS_download> Function failed
2011/11/08 16:46:44 CTS2045I R Download Managed System ldapip process
failed (rc = EOF) 2011/11/08 16:46:44 CTS1309E T CS_dwnld_ess_definitions:
Service failed. Message D774190558000004OGS01SailPoint IdentityIQ-PS130000000000
```

Workaround: Correct the container name in the application definition, save the application and run the aggregation again.

Known/Open issues

Section 2: Provisioning Integration Mod-ules

IdentityIQ can be configured to integrate with provisioning providers to automate access management for your implementation. Provisioning providers can be configured to communicate user and account information and automatically add or revoke access.

This section contains the information on the following:

Chapter 3: SailPoint IdentityIQ BMC Identity Management Suite Provisioning Integration Module

This chapter provides the steps for the installation and configuration of Identity Management Suite as an application in IdentityIQ for compliance and provisioning activities.

The following topics are discussed:

Overview

The SailPoint IdentityIQ BMC Identity Management Suite Provisioning Integration Module (IdM Suite PIM) enables IdentityIQ users to interact with the BMC Identity Management Suite.

The following table displays a comparison between terminology used in IdentityIQ and similar objects in BMC Identity Management.

BMC Identity Management	IdentityIQ
Person	Identity
Account	Account
Group	An Account Group for the IdentityIQ applications created to correspond to the BMC IdM Managed System.
Profile	An Account Group for the primary BMC Identity Enterprise SecurityStation Server IdentityIQ application.

This section explains steps that need to be performed to configure the BMC Identity Management Provisioning Integration Module to communication with BMC Identity Open Services (referred as Open Services) to perform aggregation and provisioning operations.

Supported features

The BMC Identity Manager Provisioning Integration Module provides the ability to provision BMC IdM users, Target Application accounts and entitlements from IdentityIQ.

The Provisioning Integration Module supports the following functions:

- Aggregating BMC Identity Manager Users, Groups and Target Application accounts and groups.

  Note: The BMC Identity Manager Provisioning Integration Module Supports aggregation using both the Proxy Application created for the BMC Identity Manager Provisioning System and the child Application created for each Target Application.
- Create/Update/Delete BMC Identity Manager Users
- Create/Update/Delete Target Application Accounts
- Enable/Disable/Unlock/Reset Password for BMC Identity Manager Users
- Enable/Disable/Unlock/Reset Password for Target Application Accounts
- Add/Remove BMC Identity Manager User entitlements
- Add/Remove Target Application account entitlements
- Pass through Authentication using the BMC Identity Manager System user credentials

Supported platforms

The required environment details for BMC Identity Management Suite version 7.5.01.100 or above.

Pre-requisites

To get the granular results for password change requests on selected BMC Identity Enterprise SecurityStation person and/or its connected managed systems accounts, the minimum required version of BMC Identity Management Open Services is 7.5.01.106.

Note: If you are a BMC Identity Management Suite customer with BMC Identity Enterprise SecurityStation environment you can configure BMC Identity Management Suite for **SailPoint IdentityIQ** without enabling common UI login by using the helper script provided under **iiqIntegration-BMCIdM.zip** file**.**

Configuration

This section describes the general and the operation specific configurations.

Configuring the BMC Identity Management Suite Provisioning Integration Module

BMC Identity Management Suite can be deployed on JBoss , WebLogic, or WebSphere. To get the versions of these Application Servers supported see the *BMC Identity Management Suite Guide*. To configure the BMC Identity Management Suite Provisioning Integration Module, perform the configuration steps described in the following sections:

- BMC Identity Management Suite on JBoss
- BMC Identity Management Suite on Weblogic
- BMC Identity Management Suite on WebSphere

BMC Identity Management Suite on JBoss

The following deployment scenarios are supported when BMC Identity Management Suite is deployed on JBoss:

- IdentityIQ is also deployed on JBoss
- IdentityIQ is deployed on Tomcat

Note: Only non SSL communication is supported for data exchanged between IdentityIQ and BMC Identity Management Suite that is deployed on JBoss.

IdentityIQ deployed on JBoss

Note: Ensure that IdentityIQ is configured to run on JBoss as documented in the *IdentityIQ Installation Guide*.

This section describes the configuration of BMC Identity Management Suite and IdentityIQ deployed on JBoss (for Non-SSL communication).

Perform the following steps to allow communication between IdentityIQ and BMC Identity Management Suite:

1. Extract the **identityiq.war** file in a temporary directory.
2. Copy the jar files mentioned at following location of BMC Identity Enterprise SecurityStation Open Services to the **WEB-INF\lib** directory of the IdentityIQ installation:
   From: **$BMC_IDM_SUITE_HOME\suite_sdk\lib\common** location :
   - idm-security.jar
   - idm-common.jar

   From: **$BMC_IDM_SUITE_HOME\suite_sdk\lib\jboss** location :
   - glue-client.jar
   - open-services-client.jar
3. Repackage the identityiq.war file and deploy.
4. For more information on repackaging the **identityiq.war** file, see "How to remake identityiq.war file" on page 40.
5. While configuring BMC Identity Enterprise SecurityStation Server application in IdentityIQ, Open Services Port should be set to the BMC Identity Management Open Services **JNDI** port. To get the BMC Identity Management Open Services JNDI port, run the **idm_tools jboss_ports** utility that is shipped with BMC Identity Management Open Services.
6. Enter all the other details and perform **Test connection**.

IdentityIQ deployed on Tomcat

This section describes the configuration of BMC Identity Management Suite deployed on JBoss and IdentityIQ deployed on Tomcat (forNon-SSL communication).

1. Extract the **identityiq.war** file in a temporary directory.
2. Copy the jar files mentioned at following location of BMC Identity Enterprise SecurityStation Open Services to the **WEB-INF\lib** directory of the IdentityIQ installation:
   From: **$BMC_IDM_SUITE_HOME\suite_sdk\lib\common** location:
   - idm-security.jar
   - idm-common.jar

   From: **$BMC_IDM_SUITE_HOME\suite_sdk\lib\jboss** location:
   - glue-client.jar
   - open-services-client.jar
3. Ensure that the jars listed below are copied in the **WEB-INF\lib** directory of the IdentityIQ installation: These files can be located in **%JBOSS_HOME%\client** folder of Open-services installation.
   - jnp-client.jar
   - jboss-javaee.jar
   - jboss-logging-spi.jar
   - jboss-client.jar
   - jboss-security-spi.jar
   - jboss-serialization.jar
   - jboss-common-core.jar
   - jboss-remoting.jar
   - concurrent.jar
   - jbosssx-client.jar
   - jboss-integration.jar
4. Repackage the identityiq.war file and deploy.
5. For more information on repackaging the identityiq.war file, see “How to remake identityiq.war file” on page 40.
6. While configuring BMC Identity Enterprise SecurityStation Server application in IdentityIQ, Open Services Port should be set to the BMC Identity Management Open Services **JNDI** port. To get the BMC Identity Management Open Services JNDI port, run the **idm_tools jboss_ports** utility that is shipped with BMC Identity Management Open Services.
7. Enter all the other details and perform **Test connection**.

For testing connection to Open Services through IdentityIQ Console

Add the **.jar** files mentioned in step 2 of “BMC Identity Management Suite on JBoss”section in the **WEB-INF/lib** directory of the IdentityIQ where it is extracted. After adding these files, set the SPHOME environment variable to the path which leads to WEB-INF directory.

You can determine the Open Services Port required to communicate to Open Services by means of **`idm_tools jboss_ports`** command on the BMC Identity Management Suite account.

For example, idm_tools jboss_ports

```
HTTP Port=8080

HTTPS Port=8084

JNDI Port=1099

RMI Port=1098
```

BMC Identity Management Suite on Weblogic

The following deployment scenarios are supported when BMC Identity Management Suite is deployed on WebLogic:

- IdentityIQ is also deployed on WebLogic
- IdentityIQ is deployed on Tomcat

Note: SSL as well as non SSL communication is supported for data exchanged between IdentityIQ and BMC Identity Management Suite that is deployed on WebLogic.

IdentityIQ deployed on Weblogic

Using a SSL communication channel between IdentityIQ and BMC Identity Management Suite

1. Extract the **identityiq.war** in a temporary directory.
2. Copy the following jar files from BMC Identity Management Suite installation to **\WEB-INF\lib** folder of **identityiq.war**.
   - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/common/idm-common.jar**
   - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/common/idm-security.jar**
   - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/weblogic/glue-client.jar**
   - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/weblogic/open-services-client.jar**
   - (**Only for Weblogic 10.0.2.0**) **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/weblogic/license.bea**
3. (**Only for Weblogic 10.0.2.0**) Copy **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/weblogic/license.bea** to the **WEB-INF/classes** directory of the **identityiq.war** file.
4. Repackage the **identityiq.war** and redeploy.
   For more information on repackaging the identityiq.war file, see "How to remake identityiq.war file" on page 40.
5. While configuring BMC Identity Enterprise SecurityStation Server application in IdentityIQ, Open Services Port should be **SSL listen Port** of BMC Identity Management Suite Managed server.
6. Select the **Secured connection**.
7. Provide the keystore at the location accessible by IdentityIQ managed server.
8. Enter all the other details and perform **Test connection**.

Note: BMC Identity Management Suite installer imports wl_ssl trustedCertEntry in the keystore while installing. If you are creating new keystore, ensure that the wl_ssl trustedCert is imported. The keystore used in our environment has the following entries:

wl_ssl, Jul 11, 2011, trustedCertEntry, Certificate fingerprint (MD5): A2:18:4C:E0:1C:AB:82:A7:65:86:86:03:D0:B3:D8:FE

control-sa_client, Jul 11, 2011, PrivateKeyEntry, Certificate fingerprint (MD5): 47:EA:1E:84:03:DD:67:E6:92:C7:90:24:2B:4B:0F:35

Using a non SSL communication channel between IdentityIQ and BMC Identity Management Suite

1. Extract the **identityiq.war** in a temporary directory.
2. Copy the following jars from BMC Identity Management Suite installation to **\WEB-INF\lib** folder of **identityiq.war.**
    - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/common/idm-common.jar**
    - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/common/idm-security.jar**
    - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/weblogic/glue-client.jar**
    - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/weblogic/open-services-client.jar**
3. Repackage the **identityiq.war** and redeploy.
   For more information on repackaging the identityiq.war file, see "How to remake identityiq.war file" on page 40.
4. While configuring BMC Identity Enterprise SecurityStation (ESS) Server application in IdentityIQ, Open Services Port should be **listen Port** of Suite Managed server.
5. Enter all the other details and perform **Test connection**.

IdentityIQ deployed on Tomcat

This section describes the configuration of BMC Identity Management Suite deployed on Weblogic and IdentityIQ deployed on Tomcat.

Using a SSL communication channel between IdentityIQ and BMC Identity Management Suite

1. Extract the **identityiq.war** file in a temporary directory.

2. Copy the following jar files from BMC Identity Management Suite installation to **\WEB-INF\lib** folder of **identityiq.war** file.
   - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/common/idm-common.jar**
   - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/common/idm-security.jar**
   - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/weblogic/glue-client.jar**
   - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/weblogic/open-services-client.jar**
   - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/weblogic/ wlfullclient-noServlet.jar** (For more information on **wlfullclient-noServlet.jar** file, see "JAVA File Surgery" on page 36.)
   - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/weblogic/cryptoj.jar**
   - **${BMC_IDM_SUITE_HOME}/ suite_sdk/lib/weblogic/weblogic.jar** (For more information on **weblogic.jar** file, see "JAVA File Surgery" on page 36.)
   - **${BMC_IDM_SUITE_HOME}/ suite_sdk/lib/weblogic/webserviceclient+ssl.jar**
3. Repackage the **identityiq.war** and redeploy.
   For more information on repackaging the **identityiq.war** file, see "How to remake identityiq.war file" on page 40.
4. While configuring BMC Identity Enterprise SecurityStation Server application in IdentityIQ, Open Services Port should be **SSL listen Port** of BMC Identity Management Suite Managed server.
5. Select **Secured connection**.
6. Provide the keystore at the location accessible by Tomcat on which IdentityIQ is deployed.
7. Enter all the other details and perform **Test connection**.

   Note: BMC Identity Management Suite installer imports wl_ssl trustedCertEntry in the keystore while installing. If you are creating the new keystore, ensure that the wl_ssl trustedCert is imported. The keystore used in our environment and has the following entries:

   wl_ssl, Jul 11, 2011, trustedCertEntry, Certificate fingerprint (MD5):
   A2:18:4C:E0:1C:AB:82:A7:65:86:86:03:D0:B3:D8:FE

   control-sa_client, Jul 11, 2011, PrivateKeyEntry, Certificate fingerprint (MD5):
   47:EA:1E:84:03:DD:67:E6:92:C7:90:24:2B:4B:0F:35

Using a non SSL communication channel between IdentityIQ and BMC Identity Management Suite

1. Extract the **identityiq.war** file in a temporary directory.
2. Copy the following jars from BMC Identity Management Suite installation to **\WEB-INF\lib** directory of **identityiq.war** file.
   - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/common/idm-common.jar**
   - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/common/idm-security.jar**
   - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/weblogic/glue-client.jar**
   - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/weblogic/open-services-client.jar**
   - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/weblogic/wlfullclient-noServlet.jar** (For more information on **wlfullclient-noServlet.jar** file, see "JAVA File Surgery" on page 36.)
3. Repackage the **identityiq.war** and redeploy.

For more information on repackaging the **identityiq.war** file, see "How to remake identityiq.war file" on page 40.

4. While configuring BMC Identity Enterprise SecurityStation Server application in IdentityIQ, Open Services Port should be **listen Port** of BMC Identity Management Suite Managed server.
5. Enter all the other details and perform **Test connection**.

Testing connection to Open Services through IdentityIQ Console

- **For SSL connection**: In case you are running IdentityIQ from Console then it is treated as stand- alone application and in order to do a test connection to open-services add the files mentioned in the "Using a SSL communication channel between IdentityIQ and BMC Identity Management Suite" on page 34 section to the WEB-INF/lib directory of the IdentityIQ where it has been extracted.

  After adding these files, set the SPHOME environment variable to the path, that leads to WEB-INF directory.
- **For Non-SSL connection**: In case you are running IdentityIQ from Console then it is treated as stand-alone application and in order to perform a test connection to Open Services add the files mentioned in "Using a non SSL communication channel between IdentityIQ and BMC Identity Management Suite" on page 35 section to the **WEB-INF/lib** directory of the IdentityIQ where it has been extracted.

  After adding these files, set the SPHOME environment variable to the path, that leads to WEB-INF directory.

JAVA File Surgery

For wlfullclient-noServlet.jar file

Note: You should create wlfullclient.jar file on Weblogic Server side if the wlfullclient.jar file does not exist. For more information, see the following link: http://download.oracle.com/docs/cd/E12840_01/wls/docs103/client/jarbuilder.html

Perform the following:

1. Rename the **wlfullclient.jar** file to **wlfullclient.zip**.
2. Expand the **wlfullclient.zip** file to a directory called **wlfullclient**.
3. Navigate to the expanded **wlfullclient** directory.
4. Navigate to the **javax** directory inside **wlfullclient**.
5. Delete the **servlet** directory and all of it's contents.
6. Re-zip the **wlfullclient** directory into a file called **wlfullclient.zip**.
7. Re-name the **wlfullclient.zip** file to **wlfullclient-noServlet.jar**.
   When viewing the final product, **wlfullclient-noServlet.jar** file in a Zip file editor ensure that at the top-level inside the zip file are the **javax** and its peers directories. It is easy to insert a top level directory called **wlfullclient** into the **.jar** file that will break it.

For weblogic.jar file

Perform the following:

1. Rename the **weblogic.jar** file to **weblogic.zip**.
2. Expand the **weblogic.zip** file to a directory called **weblogic**.
3. Navigate to the expanded **weblogic** directory.
4. Navigate to the **javax** directory under **weblogic**.

5. Delete the **servlet** directory and the contents.
6. Re-zip the **weblogic** directory into a file called **weblogic.zip**.
7. Re-name the **weblogic.zip** file to **weblogic.jar**.

BMC Identity Management Suite on WebSphere

The following deployment scenarios are supported when BMC Identity Management Suite is deployed on WebSphere:

- IdentityIQ is also deployed on WebSphere
- IdentityIQ is deployed on Tomcat

Note: Only non SSL communication is supported for data exchanged between IdentityIQ and BMC Identity Management Suite that is deployed on WebSphere

IdentityIQ deployed on WebSphere Server

Note: We need to have WebSphere 7 with FixPack 7.0.0.9.
Ensure that IdentityIQ is configured to run on WebSphere as documented in the *IdentityIQ Installation Guide*.

1. Extract the **identityiq.war** file in a temporary directory.
2. Copy the following jar files from BMC Identity Management Suite installation to **\WEB-INF\lib** directory of **identityiq.war** file.
   - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/common/idm-common.jar**
   - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/common/idm-security.jar**
   - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/websphere/glue-client.jar**
   - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/websphere/open-services-client.jar**
3. Repackage the **identityiq.war** file and deploy.
4. For more information on repackaging the **identityiq.war** file, see "How to remake identityiq.war file" on page 40.
5. Start the WebSphere Server.
6. Run the Integrated Solutions Console and login.
7. Click Applications and then select **Install New Application**.
8. Navigate to the directory in which you placed the **identityiq.war** file.
9. Click **Next**.
10. Ensure that **Distribute Application** and **Use Binary Configuration** are selected and that **Create MBeans for resources** is not.
11. Click Next until the installation is completed, the default settings do not need to be changed except the context root value is set to /identityiq. Make sure that you save your changes before leaving the installation panels.
12. After IdentityIQ is installed, configure it by selecting it from the Enterprise Applications list.
13. Click **Class loading and update detection** under the **Detail Properties** tab.

14. Ensure that a value is entered in the **Polling interval for updated files** field and that **Classes loaded with local class loader first (parent last)** and **Single class loader for application** are selected. The polling interval can be zero (0), but there must be a value in the field.
15. Click OK and save your changes.
16. Enable the filter compatibility mode on the application server instance where IdentityIQ is installed.
    a. From the WebSphere Integrated Solutions Console, click Servers and select Application server on which IdentityIQ is deployed.
    b. Under **Runtime => Container Settings**, expand **Web Container Settings** and click **Web Container** to display the Configuration tab..
    c. Under the Additional Properties list, click **Custom Properties** to display the **Custom Properties** table.
    d. Click **New** to display the General Properties panel.
    e. Enter the following values:
        - **Name: com.ibm.ws.webcontainer.invokefilterscompatibility**
        - **Value: true**
        - **Description: IdentityIQ filter Property**
    f. Click OK
    g. Click Save
17. In WebSphere Console, click on **Servers => Server Types => WebSphere Application Servers** (server on which IdentityIQ is deployed).
18. Click on Java and Process Management tab under Server Infrastructure, select process definition, select Java virtual Machine, Custom Properties under Additional properties.
19. Click on **New Button** and add the following parameter and its value:
    - **Name**: sun.lang.ClassLoader.allowArraySyntax
    - **Value**: true

    Apply and save to master configuration.
20. Set class loader policy of the server to single. For this, click on server on which IdentityIQ is deployed, set classloader policy to **Single** and Class loading mode to **Class loaded with local class loader first (parent last)**.
    Apply and save to master configuration.
21. The **WC_defaulthost** port of the server on which IdentityIQ is deployed must be defined in **Environment => Virtual Hosts => default_host => Host Aliases** list in the WAS7 console.
22. Restart the Server.
23. Continue with the deployment of IdentityIQ with Create the IdentityIQ Database and Tables.
24. While configuring BMC Identity Enterprise SecurityStation Server application in IdentityIQ, Open Services Port should be BOOTSTRAP_ADDRESS of the managed server of WebSphere on which Open Services is deployed.
25. Enter all the other details and perform **Test connection**.

IdentityIQ deployed on Tomcat Server

Pre-requisite:

- It is mandatory to use IBM's JDK 6.0 for IdentityIQ deployed on Tomcat Server.
- Set the JAVA_HOME environment variable to the IBM SDK and update the PATH environment variable according to IBM SDK pointing to the bin directory.
- Download **ejb.jar** file from the internet and copy it to the **WEB-INF/lib** directory of the **identityiq.war** file. This file is required only to use the BMC Identity Management Open Services EJB on Tomcat.

1. Deploy the **identityiq.war** file in tomcat server.
2. Copy the following jar files from BMC Identity Management Suite installation directory to **\WEB-INF\lib** directory of **identityiq.war** file.
   - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/common/idm-common.jar**
   - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/common/idm-security.jar**
   - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/websphere/glue-client.jar**
   - **${BMC_IDM_SUITE_HOME}/suite_sdk/lib/websphere/open-services-client.jar**
3. Copy the following jar files from the location **{WAS-HOME}/IBM/WebSphere/AppServer/deploy-tool/itp/plugins/com.ibm.websphere.v7_7.0.1.v<number>/wasJars** to **WEB-INF/lib** directory of the **identityiq.war** file.

   Note: The com.ibm.websphere.v7_7.0.1.v20081016 directory will change according to your environment.

   - bootstrap.jar
   - ras.jar
   - wsexception.jar
   - com.ibm.ws.emf.jar
   - org.eclipse.emf.ecore.jar
   - org.eclipse.emf.common.jar
   - idl.jar
   - ecutils.jar
   - iwsorb.jar
   - crypto.jar
   - com.ibm.ffdc.jar
   - naming.jar
   - namingclient.jar
4. Copy the following jar files to **WEB-INF/lib** directory of the **identityiq.war** file
   - **{WAS-HOME}/IBM/WebSphere/AppServer/runtimes/ com.ibm.ws.webservices.thinclient_7.0.0.jar**
   - **{WAS-HOME}/IBM/WebSphere/AppServer/runtimes/ com.ibm.ws.admin.client_7.0.0.jar**
5. Deploy **identityiq.war** file in Tomcat Server and start the server.

6. While configuring BMC Identity Enterprise SecurityStation Server application in IdentityIQ, Open Services Port should be **BOOTSTRAP_ADDRESS** of the managed server of WebSphere on which Open Services is deployed.
7. Enter all the other details and perform **Test connection**.

Testing connection to Open Services through IdentityIQ Console

Add idm-common.jar, idm-security.jar, open-services-client.jar, glue-client.jar, ejb.jar files from BMC Identity Management Suite installation directory to the **WEB-INF/lib** directory of the **identityiq.war** file where it is extracted.

Add the following lines in the **iiq.sh** file:

```
if [ -n "$CLASSPATH" ]; then
CLASSPATH="${CLASSPATH}${fileSep}"fi
WAS_ENV_VARS=""
WAS_HOME=WAS installation directory (e.g. /wasusr/IBM/WebSphere/AppServer)
JAVA_HOME=${WAS_HOME}/java
WAS_PROFILE=<WebSphere profile name>
THE_CP=${JAVA_HOME}/jre/lib
. ${WAS_HOME}/profiles/${WAS_PROFILE}/bin/setupCmdLine.sh
WAS_ENV_VARS="-Djava.ext.dirs=${JAVA_HOME}/jre/lib/ext:${WAS_EXT_DIRS}:${WAS_HOME}/plugins:${
WAS_HOME}/lib/WMQ/java/lib ${CLIENTSAS} ${CLIENTSSL}"
```

Modify the following line:

```
COMMAND_LINE="java $JAVA_OPTS SailPoint.launch.Launcher $DEBUG “"$@"
```

to

```
COMMAND_LINE="java ${WAS_ENV_VARS} $JAVA_OPTS SailPoint.launch.Launcher $DEBUG ""$@"
```

How to remake identityiq.war file

1. Extract **identityiq.war** file into a temporary directory.
2. Navigate to the temporary directory where **WEB-INF** is located.
3. Delete the **identityiq.war** file if previously existed by entering the following command:
   rm -rf identityiq.war
4. To create the war file run the following command in the terminal:
   jar -cvf identityiq.war *

Operation specific configuration

This section describes the various configurations required for the following operations:

- Aggregation
- Trusted Authentication
- Provisioning

Aggregation

As BMC CONTROL-SA data base consists of huge volumes of Identity data, consider the following points to avoid more time for aggregating the entire data:

- **Entities you are interested**: If you do not want to manage few entities, you can filter them out from being pulled into IdentityIQ. For more information, see "Parameters impacting Aggregation" on page 41.
- **Managed Systems in Enterprise SecurityStation you are interested and want to manage from** IdentityIQ: If you are interested in only few Managed Systems, you can aggregate against only those managed systems instead of BMC Enterprise SecurityStation main application. For this select the Managed System application as an input for aggregation task (This approach can also be used if agregating from main application is taking time). You can create a separate aggregation task for each Managed System and run them in parallel.

  Note: To create a separate aggregation task for each Managed System, define the following correlation rule:

  ```
  <CorrelationConfig name="ESS Account to Person Correlation">
      <AttributeAssignments>
        <Filter operation="EQ" property="user_id" value="name"/>
      </AttributeAssignments>
  </CorrelationConfig>
  ```

Parameters impacting Aggregation

The following configurable parameters might impact Aggregation:

- **INCLUDE ACCOUNTS**: (is an application attribute on BMC Identity Enterprise SecurityStation Server Application) determines if the connected accounts must be retrieved or no.
- **Automatically create applications**: (is an option on aggregation task) determines if for each account's Managed System Type an application should be created on IdentityIQ side or no.
- **Include Permissions**: (is the schema attribute on BMC Identity Enterprise SecurityStation Server Application) determines if the person is a BMC Identity Enterprise SecurityStation Server Administrator or no. If yes then the Managed System administrators connected to this person are retrieved and the Managed System administrators would get **Full Access to Managed System Entities** field information.
- **Profiles**: If **Profiles** which is a schema attribute, exists on BMC Identity Enterprise SecurityStation Server Application then profiles attached to a person are retrieved. Incase profiles are not of interest then this schema attribute can be deleted.
- **disableGhostAccounts**: application attribute determines to pull ghost accounts or not. In order not to pull ghost accounts, set the following value via the debug pages:

  <entry key="disableGhostAccounts" value="true"/>
- **statusMapValidityTime**: indicates at what time interval the status from BMC Identity Enterprise SecurityStation should be fetched for the changePassword transactions. The default value is 300000 milliSeconds.

- **disableAttributeFiltering**: specifies the set of fields to be returned to the connector during aggregation of BMC Identity Enterprise SecurityStation Application. The following application option is used to turn On/Off of this feature.

  <entry key="disableAttributeFiltering" value="false" />

  By default this is set to false.
- **disableSelfServiceManagerBean**: optimizes the password reset during password reset for Person and Accounts. The following application option is used to turn On/Off of this feature.

  <entry key="disableSelfServiceManagerBean" value="false"/>

  By default this is set to false.
- **multiColumnDelimiter**: The delimiter for multi-column attributes in BMC Identity Enterprise SecurityStation Server. The default value is '#'. If the default value does not work for an application, add a new separator into the application template as shown in the following example:

  <entry key="multiColumnDelimiter" value="<UserDefineValue>"/>

  Note: The character sequence "\n", "\r" or "\t" are not supported values for multiColumnDelimiter.
- For applications of type BMC Identity Enterprise SecurityStation Server, an option has been added to view a profile and the respective sub profiles in hierarchical. This option can be utilized by enabling the following attribute on the application:

  <entry key="hierarchicalProfileView" value="false"/>

Application filter for Aggregation

- You can avail application filter on BMC Identity Enterprise SecurityStation application type which will restrict the persons or accounts that are aggregated based on filter condition.

  For example:

  User ID.startsWith("a")

  Or

  User ID.startsWithIgnoreCase("a")

  In the above filter string,

  - **User ID** is a string type attribute/property of a person
  - **startsWith** represents the case sensitive search for a string
  - **startsWithIgnoreCase** represents the case insensitive search for a string

  If the above filter is applied on BMC Identity Enterprise SecurityStation application, it fetches all the person data starting with "a" and if applied on multiplexed or Managed System application, it fetches all the person data who has attached accounts starting with "a"

  Note: If any entity specific keyword, is provided in Iterate Search filter, in BMC Identity Enterprise SecurityStation application or the relevant child applications, (for example, Person or account) then aggregation for the other entity related to that application (for example, Profiles or group) will display an error.

  Note: Connected entities (accounts, profiles, direct persmissions and so on) are also fetched once a person record is fetched.

  For example, **User ID == "00000000"**

  If this filter is applied on BMC Identity Enterprise SecurityStation application or Managed System application, it fetches person "00000000" and his connected entities (accounts, profiles, direct persmissions and so on)

Note: - The BMC Identity Enterprise SecurityStation API's do not support the filter based aggregation on **org_parent** field, hence the persons based on **org_parent** filter/field would not be retrieved at the time of account aggregation.
- Maximum three conditions are permitted in a filter.
- Special characters in the filter string must be escaped using the Java escape sequence (that is, "\").
- Use label names from the application schema rather than native field names with special characters like "**__99__org_id**", and special characters like "**__**" are not supported.

- While aggregating against the child application, always use the native attributes in the filter string if specified.
  For example: **rss_user_name.startsWith("s") and not as Account ID.startsWith("s")**

  If child application refers the BMC Identity Enterprise SecurityStation application filter string, then the filter string must contain common attributes for child and the main application.
  For example, **upd_time** is part of BMC Identity Enterprise SecurityStation Application and of child application schema.

Delta like Aggregation approach

BMC Identity Enterprise SecurityStation Connector supports Delta like aggregation, which will pull the newly added and updated account and groups on BMC Identity Enterprise SecurityStation Application and the relevant child applications.

- Add the following to enable the delta like aggregation, in the BMC Identity Enterprise SecurityStation Application.

  <entry key="useDeltaLikeAggregation" value="true"/>

  This will also enable the delta like aggregation for all the child applications and will use the last successfull aggregation start time for the respective application. After adding this flag in the application, the first aggregation on BMC Identity Enterprise SecurityStation application and the relevant child applications, will perform full aggregation.
- If you do not want to enable delta like aggregation for a particular child application then you can set **<entry key="useDeltaLikeAggregation" value="false"/>** in that particular child application.
- If you have set any filter in the filter string, either in the BMC Identity Enterprise SecurityStation application or in child application then the filter will get preference over delta like aggregation flag.
- Delta like aggregation only pulls accounts and groups updated after the last successful aggregations start time. It will not fetch newly added or removed entitlements of accounts on the BMC Identity Enterprise SecurityStation or child applications. In addition, any account or group of the BMC Identity Enterprise SecurityStation or child application, that is deleted, will also not be captured

Note: **When using a filter string, the aggregation task's "Detect deleted accounts" option should not be used or all accounts previously aggregated which are outside of the filtered set will be deleted.**

Configuration for Aggregation

Configure IdentityIQ to aggregate accounts from BMC Identity Management Suite by creating an IdentityIQ application as follows:

1. Navigate to the IdentityIQ **Define=>Application** page.
2. Create a new application of type **BMC ESS Server**.
3. On the Attributes tab, enter the details required to connect to BMC Identity Management Open Services including the **Open Services Host**, **Open Services Port/JNDI Port** and **Open Services Host Type**.
4. To enable a secured connection to Open Services, perform the following steps:

   a. Select **Secured Connection**.
   b. Enter the **Keystore File Location** containing the Application Server Client Certificates.
   c. Provide the security provider class dependent on the J2EE Application Server. By default the following is provided:
      - **(For JBoss and Weblogic) `com.sun.crypto.provider.SunJCE`**
      - **(For IBM WebSphere) `com.ibm.crypto.provider.IBMJCE`**

5. Enter the BMC Identity Enterprise SecurityStation Administrator Name in the **Username** field and enter one of the following password options:
   a. Person password if Common UI Flag is turned On.
   b. Admin password if Common UI Flag is turned Off.

   Note: The BMC Identity Enterprise SecurityStation Administrator should have the view of all access rules configured for aggregation and all required access rules for update on person, profile, group, and accounts.
6. Click **Test Connection** to verify the connection to BMC Identity Open Services.

   Note: For more information on BMC Identity Enterprise SecurityStation Server Application Configuration, see *BMC Identity Management Suite SDK Programmer's Guide*.

Aggregating from BMC Identity Management

To aggregate BMC Identity Management persons and managed accounts, create and execute an IdentityIQ Account Aggregation task. Include the BMC Identity Enterprise SecurityStation Server application in the list of applications scan.

When the aggregation task is complete, a new application is created for every managed system in BMC Identity Management. The application schema includes attributes from the managed accounts. All persons in BMC Identity Management have an account created that is associated with the BMC Identity Enterprise SecurityStation Server application and any administrative capabilities and assigned profiles are included as attributes or direct permissions of those accounts. Additionally, all of the managed accounts, including those not associated with a person, are aggregated and associated with the newly created applications.

Once the initial aggregation from the BMC Identity Enterprise SecurityStation Server application is completed, you can aggregate from it again to read in information for all of the managed systems, or you can aggregate from the individual, automatically created applications.

Note: You can make use of the BMC Identity Enterprise SecurityStation Application Creator task to discover all of the Managed Systems present in the BMC Identity Enterprise SecurityStation environment. The input for this task is an application of type BMC Identity Enterprise SecurityStation Server. Executing this task results in the creation of all multiplexed Managed System applications. You can aggregate on these individual applications to manage only a few managed systems in your environment.

Trusted Authentication

Trusted connections allow logging in to BMC Identity Management Open Services and BMC Identity Enterprise SecurityStation with only a user name only (no need for a password).

To configure a Trusted Connection between IdentityIQ and Identity Open Services, it is necessary to export the IdentityIQ keystore certificate file, and then import the certificate file to the IdM Suite keystore.

Note: The open service supports only the JKS type of keystore. Use JDK of the Sun/IBM/Open JDK to create new keystore. The JAVA_HOME environment variable should point to one of these JDK, so the default keystore created of type JKS or use the JDK that is with the BMC Identity Management Open Services located at <BMC_IDM_SUITE_HOME>/BMCSuiteInstallJVM/ to create new keystore and copy it on IdentityIQ side.

The following section describes the procedure for creating the keystore.

Creating the keystore

Open the command prompt and use the keytool command which comes with the java.
On IdentityIQ

1. Create a new keystore by specifying the alias name, domain name, keystore name and algo name. Keep algorithm name as RSA. Provide the same keystore and alias password using the following command:
   ```
   >>keytool -genkey -alias <alias name> -keyalg RSA -dname <a full domain
   name like : CN=adminuser.sailpoint.com> -keystore <keystore name like :
   iiq.keystore>
   ```
   Note: Keystore and alias password must be same for BMC IDM Open Services.
2. To see the content of the keystore, use the following command:
   ```
   >>keytool -list -v -keystore <keystore name> -storepass <keystore
   password>
   ```
3. Extract the public key as a certificate from the keystore to share it with the client using the following Command :
   ```
   >>keytool -v -exportcert -alias <alias name> -file < certificate name like
   : publicKey.cer> -keystore <keystore name>
   ```
4. Copy this public certificate on the Open Services side.

On BMC Identity Management Open Services side

Import the public certificate in the **idm.keystore** of the open services. This keystore is located at **<BMC_IDM_SUITE_HOME>\security\keystore** directory:

```
>>keytool -importcert -v -alias <alias name> -file <public certificate name>
-keystore <keystore name like : idm.keystore> -storepass <keystore password>
```

5. In BMC Identity Enterprise SecurityStation Application, provide the details of newly created keystore for the trusted authentication.

Provisioning

The following provisioning operations are available in IdentityIQ when integrating with the IdM Suite:

- Update, Delete, Revoke, Restore, Unlock Operations on Person(s)
- Update, Delete, Revoke, Restore, Unlock Operations on Account(s)
- Connect Accounts to Person
- Connect Accounts to Group
- Connect Profile to Person

Note: User should use the same password for person and accounts. If different passwords are set for person and accounts, then only the person password is updated.

Password management

If you want to achieve the password synchronization by selecting only the Person record, add the **syncPersonAndAccountsPassword** application attribute to BMC Identity Enterprise SecurityStation application created in IdentityIQ.
For example, **<entry key="syncPersonAndAccountsPassword" value="true"/>**

To get the granular results for selected BMC Identity Enterprise SecurityStation person and/or its connected managed systems accounts, the minimum required version of BMC Identity Management Open Services is 7.5.01.106.

Add the following attribute in the application debug of the BMC BMC Identity Enterprise SecurityStation Application to get the granular status in change password operation:

<entry key="openServicesVersion" value="7.5.01.106"/>

Provisioning Retry

All unsuccessful Provisioning requests under any of the following category will be performed again from BMC Identity Management Suite Provisioning Integration Module:

- Communication and Socket exceptions while attempting to connect to BMC Identity Management Suite/ Open Services.
- When connection to BMC Identity Enterprise SecurityStation Application Server/BMC Identity Enterprise SecurityStation Database could not be established.

Following settings that can be configured on the application definition to drive specific retry behavior:

<entry key="provisioningRetryThreshold" value="10"/>   // minutes to wait for retry

<entry key="provisioningMaxRetries" value="50"/>   // number of retries to attempt before failure

Provisioning Status

The **IdentityRequest** object (LCM Manage my AccessRequest) will now hold provisioning status and result description. The end user can view the exact status of provisioning request at BMC Identity Enterprise SecurityStation Server side. Below would be the status of Requests based on ESS Transaction status.

- **Pending**: if at least one transaction from Request Items is pending at BMC Identity Enterprise SecurityStation end.
- **Committed**: if the connector has completed all of the request items in the provisioning plan.
- **Failed**: if at least one transaction from Request Items is failed at BMC Identity Enterprise SecurityStation end.

BMC Identity Enterprise SecurityStation	IdentityIQ
Waiting	Pending
Started	Pending
OK	Committed
Error	Failed
Reset	Committed
Sent	Pending

Ghost account creation

To create ghost account add the following flag in the BMC Identity Enterprise SecurityStation Application debug page and set its value to **true**:

```
<entry key="disablePersonConnection" value="true"/>
```

Note: **To create aghost account, do not make User ID or user_id as a mandatory field in the child or managed system application's provisioning policies.**

Add Connections

If **disablePersonConnection** attribute of BMC Identity Enterprise SecurityStation application is set to **true** and in IdentityIQ, if we try to connect an identity with no BMC Identity Enterprise SecurityStation person and no managed system account to a manged system group then IdentityIQ fails. We need to create a BMC Identity Enterprise SecurityStation person first.

Entitlement

- Identity with no BMC Identity Enterprise SecurityStation person connected to it and single/multiple managed system groups connection is not supported.
- Identity with no BMC Identity Enterprise SecurityStation person connected to it and single/multiple BMC Identity Enterprise SecurityStation profiles connection is supported, provided the following condition is satisfied:

  The template is defined for managed system account creation on BMC Identity Enterprise SecurityStation Server side, so that groups connected to the BMC Identity Enterprise SecurityStation profile would get connected to the newly created managed system account.

Troubleshooting

This section provides the resolutions for the following errors that may be encountered while setting up and configuring BMC Identity Management Suite.

For debugging issues with Aggregation and Provisioning, **enable log4j** logging on the **sailpoint.connector.ctsa** package class. The following are sample lines for the log4j.properties configuration file:

```
log4j.logger.sailpoint.connector.ctsa=debug
```

1 - An error is displayed when trying to connect to BMC Identity Management Suite application on Weblogic

The following error is displayed when trying to connect to your BMC Identity Management Suite application on Weblogic:

Throwing createConnector - java.lang.NoClassDefFoundError: javax/ejb/CreateException.

The above error is displayed as the **.jar** file surgery was not performed correctly.

Resolution: Open the **.jar** files in a **.zip** file editing tool. If the **.jar** file contains a folder with the same name as that of the **.jar** file then it is incorrect. The top-level directories in the **.jar** file should be **org** or **javax** and not the name of the **.jar** file.

WARNING—It is a common mistake to insert an extra directory at the top of a .jar file when re-zipping the expanded .jar file back into a .zip file.

2 - IdentityIQ connections causing Open Services to Dump Core

There have been instances where IdentityIQ connecting to the Open Services BMC Identity Enterprise SecurityStation has caused the Open Services application server to fail (dump core).

Weblogic Application Server instances on which Identity Management Suite (IDM) is deployed sometimes crashes when trying to communicate with BMC Identity Enterprise SecurityStation Server if Database or BMC Identity Enterprise SecurityStation Application Server is not responding.

Resolution: Perform the following:

1. Try logging into the BMC Identity Management Suite UI with the same account you are using in IdentityIQ. In ESS API logs we can get to know for what reason suite is waiting.

   Note: You have to enter the person password in order to logon to BMC Identity Management Suite.

2. Try logging into the BMC Enterprise SecurityStation Console with the same user name and password as that of IdentityIQ.
3. Weblogic thread dumps and ESS API logs (in location mentioned above) can help us identify the problem over here.

Once the Open Services problem is identified, tune the database or BMC Identity Enterprise AppServer parameters.

3 - SSL Certificate Issue

When using SSL to communicate with the Open Services server the IdentityIQ Application Server should have a complete trust chain to the open services server. This means that the IdentityIQ Server must be configured with a key store that trusts the organization's CA (Certificate Authority) host and have the full chain down to the Open Services host's certificate and public key.

If the CA is incorrectly configured the following error message is displayed:

Problem creating context [javax.naming.CommunicationException [Root exception is java.net.ConnectException: t3s://opensvcshost.somecompany.com:8002: Destination unreachable; nested exception is: javax.net.ssl.SSLKeyException: [Security: 090542] Certificate chain received from opensvcshost.somecompany.com - 10.193.10.6 was not trusted causing SSL handshake failure. Check the certificate chain to determine if it should be trusted or not. If it should be trusted, then update the client trusted CA configuration to trust the CA certificate that signed the peer certificate chain. If you are connecting to a WLS server that is using demo certificates (the default WLS server behaviour), and you want this client to trust demo certificates, then specify

-Dweblogic.security.TrustKeyStore=DemoTrust on the command line for this client.;

No available router to destination]]

Resolution: Temporarily override the trust chain for the open services SSL certificate on Weblogic servers that use the **demo** certificates by specifying the following configuration in Java environment:

-Dweblogic.security.TrustKeyStore=DemoTrust

Note: The above configuration cannot be used to turn off the trust chain verification.

To permanently disable the SSL trust chain by starting your Java environment with the following parameter which should never be used:

-Dweblogic.security.SSL.ignoreHostnameVerification=true

This leaves the SSL system on the server open to man-in-the middle attacks as the IdentityIQ application server never verifies through the trust chain that the Open Services service is correct and presenting the correct public encryption key. The easiest way to deal with this is to import the correct trusted CA information.

4 - While performing the test connection with trusted authentication the “Invalid keystore format’ error is displayed.

If BMC Identity Management Suite Provisioning Integration Module is configured for the trusted authentication with the BMC Identity Management Open Services, and in the test connection if the following error is displayed, it means the keystore created are not of JKS type:

Invalid keystore format

Resolution: To resolve this issue, perform one of the following:

- use JDK of Sun/IBM/Open JDK which will create the new keystore of type JKS.

use the JDK which comes with the BMC Identity Management Open Services located at **<BMC_IDM_SUITE_HOME>/BMCSuiteInstallJVM** and copy the new keystore on IdentityIQ side.

Chapter 4: SailPoint IdentityIQ Oracle Identity Manager Provisioning Integration Module

The following topics are discussed in this chapter:

Overview

This chapter provides a guide to the integration between Oracle Identity Manager (OIM) and IdentityIQ. This chapter is intended for Oracle and IdentityIQ System Administrators and assumes a high degree of technical knowledge.

The integration is achieved by deploying a small web application in the application server that hosts OIM. IdentityIQ communicates with the web services contained in this application to read and write account information. Configuration of the OIM integration requires the username and password of the OIM administrator or another user with sufficient permissions.

Supported features

The Oracle Identity Manager Provisioning Integration Module supports the following functions:

- Aggregating Oracle IdM Users and Target Application accounts.

  Note: The Oracle Identity Manager Provisioning Integration Module Supports aggregation only using the Proxy Application created for the Oracle IdM Provisioning System. Does not support aggregating groups

- Create/Update/Delete of Oracle Identity Manager User

  Note: Creating Oracle Identity Manager is only supported from Oracle Identity Manager 10 and a*bove*

- Enable/Disable/Unlock of Oracle Identity Manager User
- Add/Remove Oracle Identity Manager User Entitlements

  Note: Add/Remove Oracle IdM user entitlements is supported from Oracle Identity Manager 10 and above

- Create/Update/Delete of Target Application accounts
- Enable/Disable/Unlock of Target Application accounts
- Add/Remove Target Application accounts Entitlements

Supported platforms

- Oracle Identity Manager 9 (Oracle 10g)
- Oracle Identity Manager 11g R1

Installing the OIM Integration Web Application

You must first deploy the OIM Integration Servlet web application to the application server hosting the OIM application. The **iiq.war** file for this web application is contained in the IdentityIQ distribution as $INSTALLDIR**/integration/OIM/iiqIntegration-OIM.jar** or in the distribution for an IdentityIQ patch in a **.jar** file named **Integration-oim-<version>.jar**.

The **iiqIntegration-OIM.jar** file contains **iiq.war** file. You can customize the **iiq.war** file in many ways before being deployed into the application server hosting OIM.

Note: Ensure that if you are deploying web application as war file, it should be named as iiq.war. If you are deploying the web application from a directory, then directory should be named as iiq.

The following are the required customization steps:

1. Configure access to OIM by modifying **WEB-INF/classes/xellerate.properties** to set.
   - **XL.HomeDir**: the full path to the directory where OIM is installed
   - **userName**: the OIM administrator that has the appropriate permission to read and write user and account data
   - **password**: the password for the OIM administrator

   For more information on the other properties that need to be set in **xellerate.properties** file, see "Properties that can be defined in xellerate.properties" on page 54.
2. Copy the required API implementation .jar files from the OIM installation into the **WEB-INF/lib** directory of the integration application:
   - (**For Oracle 11g**) Copy **OIM_ORACLE_HOME/designconsole/lib/oimclient.jar**
   - (**For Oracle 10g**)
     - Copy the following files from the OIM web lib directory to **iiq.war** only if **iiq.war** and OIM server are installed on separate WebLogic managed server:
       - **javagroups-all.jar**
       - **oscache.jar**
       - **wlXLSecurityProviders.jar**
       - **XIMDD.jar**
       - **XL10SecurityProviders.jar**

- **Copy the following files from the OIM external lib directory**
  - **xlAdapterUtilities.jar**
  - **xlAPI.jar**
  - **xlAttestation.jar**
  - **xlAuditor.jar**
  - **xlAuthentication.jar**
  - **xlBackOfficeBeans.jar**
  - **xlBackofficeClient.jar**
  - **xlCache.jar**
  - **xlCrypto.jar**
  - **xlDataObjectBeans.jar**
  - **xlDataObjects.jar**
  - **xlDDM.jar**
  - **xlGenConnector.jar**
  - **xlGenericUtils.jar**
  - **xliGCProviders.jar**
  - **xlInputPreprocessor.jar**
  - **xlInstaller.jar**
  - **xlLogger.jar**
  - **xlRemoteManager.jar**
  - **xlRequestPreview.jar**
  - **xlSampleApp.jar**
  - **xlScheduler.jar**
  - **xlUtils.jar**
  - **xlVO.jar**
  - **xlWebClient.jar**
  - **xlWSCustomClient.jar**

Testing the OIM Integration Web Application

Verify if the installation was successful using the following steps:

Note: For each test URL throughout this document, change the host name and port to match your OIM Server instance.

1. From any browser enter the following URL:
   http://localhost:8080/iiq/resources/ping

   The following response is displayed:

   OIM integration ready

   Failure to get a ping response indicates a problem with the deployment of the servlet.

2. Verify the integration servlet can communicate with OIM by entering the following URL:
   http//localhost:8080/iiq/resources/users

   You should see a response containing the names of all OIM users. This might take a while to assemble depending on the number of users. To view details of a particular user, enter the following URL where <*OMIUSER*> is the name of a user in your OIM instance:

   http://localhost:8080/iiq/resources/user/<OIMUSER>

To see additional diagnostic information for of a particular user, enter the following URL where <*OMIUSER*> is the name of a user in your OIM instance:

http://localhost:8080/iiq/resources/debug/<OIMUSER>

If you are unable to request user information, there may be a problem with the credentials you entered in the **xellerate.properties** file. For more information, see "Properties that can be defined in xellerate.properties" on page 54.

Properties that can be defined in xellerate.properties

1. Add a ManagedResource definition in the ManagedResource list for an each OIM resource. For each resource, define a property prefix by adding a property whose name is the prefix and whose value is the OIM resource name.

   For example:

   AD=AD User

   Oracle=Oracle DB User

   This declares that any property that begins with ERP is related to the OIM resource named ERP Central Component.

2. For each ManagedResource, define the account attribute that represents the unique account identifier. The names used here must be the resource names used by OIM. The identityAttribute must have the internal form field name containing the account identifier. Use the OIM Design Console application to find the process form for each resource and view the field names. The example below gives two typical names, one used by the connector for Oracle database users and the other for the Active Directory connector.

   AD.id=UD_ADUSER_UID

   Oracle.id=UD_DB_ORA_U_USERNAME

3. Define the names of the child forms that support multiple attributes. The value is a CSV of the internal child form names:

   AD.childForms=UD_ADUSRC

   Oracle.childForms=UD_DB_ORA_R

   In this example **UD_ADUSRC** is the internal name for the child form AD User Group Details and **UD_DB_ORA_R** is the internal name for the child form DBUM Grant/Revoke Roles.

4. Each child form name in the Oracle.childForms property there is another property whose value is a CSV of the child form fields to return and the order in which they will appear in IdentityIQ.

   Oracle.UD_DB_ORA_R=UD_DB_ORA_R_ROLE,UD_DB_ORA_R_ADMIN_OPTION

   In the previous example, we will return two fields from the child form UD_DB_ORA_R. The first field has the Role name and the second has the Role Admin option.

5. Following is the configuration for resource with child forms: ERP Central Component:

   ERP=ERP Central Component

   ERP.id=UD_ECC_USER_ID

   ERP.childForms=UD_ECC_PRO,UD_ECCRL

   ERP.UD_ECC_PRO=UD_ECC_PRO_SYSTEMNAME,UD_ECC_PRO_USERPROFILE

   ERP.UD_ECCRL=UD_ECCRL_SYSTEMNAME,UD_ECCRL_USERROLE

Note: Before **IdentityIQ** 6.0 there was a parameter in **xellerate.properties** file as **oldChildFormNames** which was used for the resources who have only one field in the childform, for example, Active Directory resource. For **IdentityIQ** version 6.0 onwards, the value must be set to true if the user wants to support **oldChildFormNames** where field returned would be form name + field name (For example, **UD_ADUSRC:UD_ADUSRC_GROUPNAME** field in Active directory).

6. To aggregate all the active and disabled OIM users in IdentityIQ, add a new parameter **OIM_USER_TYPE** in **xelerate.properties** file with the value as **ALL**. If **OIM_USER_TYPE** parameter is removed from the **xeler-ate.properties** file then only the active OIM users will be aggregated. By default only active OIM user are aggregated.

Configuration for OIM application

Perform the following steps to create an IdentityIQ application for OIM:

1. Navigate to the IdentityIQ **Define=>Application** page.
2. Create a new application of type **Oracle Identity Manager**.
3. On the Attributes tab, enter the Oracle Identity Manager Host and Oracle Identity Manager Port.
4. Click **Test Connection** to verify the connection to OIM.

Note: **You can make use of the "OIM Application creator" task to discover all the resources present in OIM environment. The input for this task would be an newly created application of type "Oracle Identity Manager" and executing this task would result in the creation of all multiplexed resources.**

Note: **After upgrading to IdentiyIQ 6.1, the application type of the application created with application type OIM, will be changed to Oracle Identity Manger .**

Testing the OIM Integration Client

While any IdentityIQ feature that generates a provisioning request such as a certification remediation, a role assignment, or a Lifecycle Manager request can be used to test the integration, it is sometimes useful to test at the provisioning layer using the IdentityIQ integration console.

Launch the console by using the IdentityIQ script in the **INSTALLDIR/WEB-INF/bin** directory of the IdentityIQ installation to run **iiq integration**.

From the console command prompt, use the **list** command to display the names of all Application objects created in the system. Using the example in the previous section, verify an Application object of type **Oracle Identity Manager** exits.

Use the following command:

use *OIMApplicationName*

Use the **ping** command to initiate a test connection message with OIM. A successful connection will return the following message:

Response: Connection test successful

If any problem occurs in the communication of this application with the OIM Integration Web Application, troubleshoot this application by viewing the application server logs for both the IdentityIQ and OIM application servers. You can enable log4j tracing on both sides by using the following:

log4j.logger.iiq.integration=debug

log4j.logger.iiq.connector=debug

This lets you see if the requests are transmitting over the network, and how they are processed.

If the OIM servlet is deployed on Weblogic 11g, tracing can be enabled on it by adding an entry to the logging file on the Weblogic server. Following is the logging file:

<DOMAIN_HOME>/config/fmwconfig/servers/oim_server1/logging.xml

Following is the entry that needs to be added:

<logger name="SailPoint.integration.oim" level="TRACE:32"/>

For more information on enabling system logging in OIM is included in the *Oracle Identity Manager Administrator Guide*.

Aggregating from OIM

To aggregate OIM users and resource accounts, create and execute an IdentityIQ Account Aggregation task. Include the OIM application in the applications to scan list.

When the aggregation is complete from the OIM application, a new application is created for every resource in OIM. The application schema includes attributes seen in the resource accounts. All users in OIM have an account created that is associated with the OIM application and includes all of the standard and extended user attributes of those users. Additionally, all of the resource accounts are aggregated and associated with the newly created applications.

Once the initial aggregation from the OIM application is completed, you can aggregate from it again to read in information for all managed systems.

Note: **You can make use of the “OIM Application Creator task” to discover all of the Resources present in the OIM environment. The input for this task is an application of type Oracle Identity Manager. Executing this task results in the creation of all multiplexed Resource applications.**

Known/Open issues

Following are the known/open issues of Oracle Identity Manager:

- Creating Oracle Identity Manager user does not work with Oracle Identity Manager 9.1.0.
- You cannot perform provisioning operations simultaneously on the OIM server from IdentityIQ and the OIM console. This is a class loading issue observed with OIM 11g, after deploying iiq servlet(iiq.war) on Weblogic OIM Managed Server.

  Workaround for this issue: Create another, empty WLS(Weblogic)Managed server next to the OIM Managed Server and only deploy the IIQ Servlet. Also, update the Xellerate.properties file by un-commenting the attribute "java.naming.provider.url". This Url needs the host name of the host where OIM managed server is deployed and the listening port of the OIM managed server.

Chapter 5: SailPoint IdentityIQ Sun Identity Manager Provisioning Integration Module

The following topics are discussed in this chapter:

Introduction

The integration between IdentityIQ and Sun Identity Manager is very straightforward and requires little configuration on either side to start the initial communication. This document covers the important points for configuring an integration between these products.

Supported features

The Sun Identity Manager Provisioning Integration Module provides the ability to provision Sun Identity Manager users, Target Application accounts, groups and entitlements from IdentityIQ.

The Provisioning Integration Module supports the following functions:

- Aggregating Sun Identity Manager Users and Target Application accounts.

  Note: The Sun Identity Manager Provisioning Integration Module Supports aggregation only using the Proxy Application created for the Sun Identity Manager Provisioning System. Does not support aggregating groups

- Create/Update/Delete Sun Identity Manager Users
- Enable/Disable/Unlock/Reset Password for Sun Identity Manager Users
- Add/Remove Sun Identity Manager User Entitlements
- Create/Update/Delete Target Application Account
- Enable/Disable/Reset Password for Target Application Accounts
- Add/Remove Target Application Account Entitlements

Supported platforms

- Oracle Waveset 8.1.1
- Sun Identity Manager 8.1
- Sun Identity Manager 8.0
- Sun Identity Manager 7.1
- Sun Identity Manager 7.0

General configuration

IdentityIQ communicates with Sun Identity Manager using a Service Provisioning Markup Language (SPML) interface. To enable this communication, the SPML client library, **openspml.jar**, must be copied from the Sun IdM installation into the **INSTALLDIR/WEB-INF/lib** directory of the IdentityIQ installation and then the IdentityIQ application should be restarted. A system error will occur in IdentityIQ when defining a Sun Identity Manager application before this configuration step is completed.

Configuration for Aggregation

Configuring IdentityIQ to aggregate accounts from Sun Identity Manager is accomplished by creating an IdentityIQ application as outlined in the following steps:

Configure SPML in Sun Identity Manager:

1. If you do not have a Configuration:SPML object in Sun Identity Manager, import **sample/spml.xml** from the Sun Identity Manager installation.
2. Add the following object into the SPML classes list:

```
  <Object name='IIQUserView'>
    <Attribute name='type' value='User'/>

    <!-- keep the default form quiet -->
    <Attribute name='form'      value='view'/>
    <Attribute name='viewForm'   value='Empty'/>
    <Attribute name='identifier' value='waveset.accountId'/>
    <Attribute name='default'    value='true'/>

<!-- May need this if the install ordinarily turns on the meta-view -->
    <Attribute name='viewOptions'>
      <Object>
        <Attribute name='ApplyMetaView' value='false'/>
      </Object>
    </Attribute>

<!-- This is important for ModifyRequests to remove selected elements.
    We'll have to put something here for every multi-valued attribute
   in every managed resource.  Tedious but at this point necessary.
    <Attribute name='multiValuedAttributes'>
      <List>
        <String>IDM/IdentityIQ Integration Demo::groups</String>
        <String>accounts[IDM/IdentityIQ Integration Demo].groups</String>
      </List>
    </Attribute>
    -->
  </Object>
```

Define IdentityIQ Application:

1. The object name (in this case IIQUserView) should match the Native Object Type of the account schema IdentityIQ application that represents Sun Identity Manager.

2. When creating the Sun Identity Manager application in IdentityIQ, follow the default pattern for the rpcRouterURL to point to the Sun Identity Manager system.

To aggregate Sun Identity Manager users and resource accounts, create and execute an IdentityIQ Account Aggregation task for the Sun Identity Manager application.

When the aggregation task is complete, a new application is created for every managed resource in Sun Identity Manager. The application schema includes attributes from the resource accounts. All users in Sun Identity Manager have an account created that is associated with the Sun Identity Manager application and any administrative capabilities and assigned profiles are included as attributes or direct permissions of those accounts.

Once the initial aggregation from the Sun Identity Manager application is completed, you can aggregate from it again to read in information for all of the resource systems.

Configuration for Provisioning

No additional configuration is needed to enable provisioning to Sun Identity Manager. The Sun Identity Manager PIM is implemented using the latest IdentityIQ connector interface that provides read/write capability. If you wish to provision accounts for Sun Identity Manager applications using another method, then you must remove the **PROVISIONING** keyword from the **featureString** in the Sun Identity Manager application in IdentityIQ.

Chapter 6: SailPoint IdentityIQ Novell Identity Manager Provisioning Integration Module

The following topics are discussed in this chapter:

Overview

This chapter provides a guide to the Novell® Identity Manager and SailPoint IdentityIQ integration and configuration for your enterprise. This chapter is intended for Novell and IdentityIQ System Administrators and assumes a high degree of technical knowledge of these systems.

Supported features

The Novell Identity Manager Provisioning Integration Module provides the ability to provision Novell Identity Manager users, Target Application accounts and entitlements from IdentityIQ.

The Provisioning Integration Module supports the following functions:

- Create/Update/Delete Novell IdM User
- Create/Update/Delete Target Application Account
- Enable, Disable Novell IdM Users
- Enable/Disable Target Application Accounts
- Add/Remove entitlements for Target Application Account
- Aggregating Novell IdM Users and Groups.

  Note: While aggregating Novell IdM Users/Groups, the connected target system Accounts/Groups are also aggregated.

Note: Before you can use the provisioning feature of the connector, all PRDs must be defined and configured on Novell side. For more information about the PRD's, see “IdentityIQ Provisioning workflows” on page 63 **section.**

Supported platforms

- Novell Identity Manager 3.0
- Novell Identity Manager 4.0

Pre-requisites

It is mandatory to have the user application module installed on the Novell Identity Manager.

Configuring Novell Identity Manager for IdentityIQ Integration

The integration has dependencies on the following Novell Identity Manager components:

- Connected Systems Drivers
- Role Based Entitlements (RBEs)
- User Service Driver
- IdentityIQ Provisioning Workflows
- Enabling the Web Service

Connected systems

The DirXML drivers, connected to the Identity Vault from which the data is aggregated, have to be mapped to IdentityIQ Applications. This is done using the Novell Application Generator task (see below).

Role Base Entitlements (RBEs)

Role Based Entitlements (RBEs) need to be defined in the Identity Vault before automatic remediation from IdentityIQ can occur. Policies can be setup in the vault to add and remove account membership, group membership, etc. when these RBEs are granted or revoked. When accounts are aggregated into IdentityIQ, these RBEs show up as IdentityIQ entitlements.

User Service Driver configuration

The user service driver provides the provisioning and directory abstraction layer (VDX) web service that IdentityIQ uses to communicate with Novell Identity Manager to perform remediation and aggregation respectively. It is assumed that the user service driver is the Entitlement granting authority.

Note: A conflict occurs if you have an Entitlement Service Driver for the same driverset.

IdentityIQ Directory entities

Directory entities must be defined for the User Service driver that are used by IdentityIQ when it invokes the VDX web service. This entity is defined for the User Service driver under Directory Abstraction Layer.

For Account aggregation, entity must be created with the details mentioned in the following table:

Task	Entity Key	Object Class	Attributes
For Novell Application Creator Task	dirXML-Driver	DirXML-Driver	CN (allow search access to users), Object Class (allow search access to users)
	For Novell 4.*x* the following entity is also required		
	DirXML-Entitlement	DirXML-Entitlement	CN (allow search access to users), Object Class (allow search access to users). Provide value for the Search Container
For Account Aggregation	user	User	CN, Department, Direct Reports, DirXML-Accounts, DirXML-Associations, DirXML-EntitlementRef, Email, First Name, Group, Hidden attribute List, Last Name (allow search access to users), Login Disabled, Manager, Preferred Notification, Query List, Region, Telephone Number, Title
For Account Group Aggregation	group	Group	CN, Description (allow search access to users), DirXML-Associations, DirXML-EntitlementRef, Members

Note: - For entity **DirXML-Driver**, the attributes CN and objectClass are mandatory and they should be marked as Searchable.
- For **user** and **group** entities, the attribute keys should be in sync with account and group schema attribute names in IdentityIQ.
- For user entity, the **LastName** attribute is mandatory and it should be marked as Searchable.
- For group entity, the **Description** attribute is mandatory and it should be marked as Searchable.

Save these entities and import into the identity vault. Note the key of these entities as it is used in IdentityIQ.

IdentityIQ Provisioning workflows

Provisioning workflows must be defined for the User Service driver that are used by IdentityIQ when it invokes the provisioning web service. This workflow is defined for the User Service driver under Provisioning Request Definitions. You can find sample workflows (PRD) under integration/nidm/exampleWorkflows. Import them under Provisioning Request Definitions. After importing add a valid object in Trustee rights. Sample workflows (PRD) for the corresponding operations are as described in the following table:

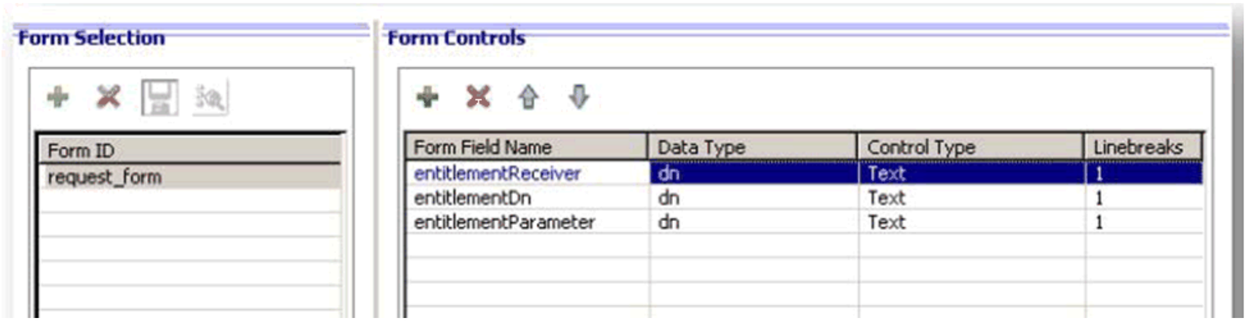
PRD	Operation	Form fields
SPCreateModifyUser.xml	Create and Update Vault User	FirstName, LastName, Title, Location, Department, Email, loginDisabled, manager, group

PRD	Operation	Form fields
SPDeleteUser.xml	Delete Vault User	Form fields are not required for this workflow.
SPDisableEnableUser.xml	Enable and Disable Vault User	loginDisabled
SPEntitlements.xml	Request Access for Vault User	entitlementReceiver, entitlementDn, entitlementAction, entitlementParameter

Note: Additional workflow steps and logic might be required based on your business needs.
- For Create and update user workflow, the field names and number of fields in provisioning policy on IdentityIQ side should be same as that of the input field names of workflow.
- For Enable/Disable vault user workflow and Request access workflow (that is, Entitlement), the names of input parameters of workflow should be same as given in above table. Also all mandatory fields on Novell side should be marked as "required" in IdentityIQ provisioning policy.

To define a workflow with the minimum required steps, perform the following:

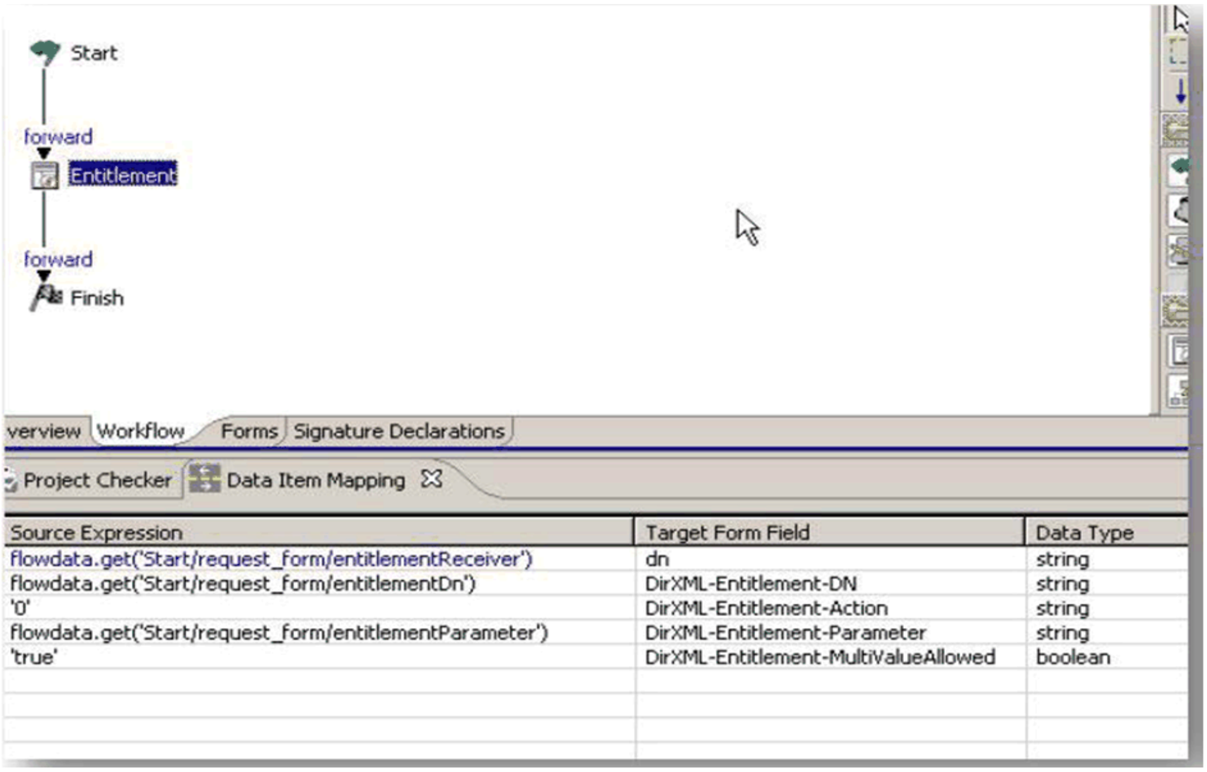
1. Create a provisioning request definition of type Accounts/Entitlements.
2. Define a simple workflow with three steps, Star, Entity/Entitlement and Finish.
3. Define a form with fields as shown in the Sample PRD below:



Form ID
request_form

Form Field Name	Data Type	Control Type	Linebreaks
entitlementReceiver	dn	Text	1
entitlementDn	dn	Text	1
entitlementParameter	dn	Text	1

4. Verify in the Entity/Entitlement step of the workflow that the Source expressions are defined correctly.



Source Expression	Target Form Field	Data Type
flowdata.get('Start/request_form/entitlementReceiver')	dn	string
flowdata.get('Start/request_form/entitlementDn')	DirXML-Entitlement-DN	string
'0'	DirXML-Entitlement-Action	string
flowdata.get('Start/request_form/entitlementParameter')	DirXML-Entitlement-Parameter	string
'true'	DirXML-Entitlement-MultiValueAllowed	boolean

5. Save this workflow and import into the identity vault. Note the dn of this workflow as it is used in defining the Novell Application attributes later in the process.

Enabling the Web Service

The integration uses the following Web Service endpoints provided:

- Directory Abstraction Layer (VDX) Web Service for aggregation
- Provisioning Web Service for remediation

By default, the endpoints are disabled. To provide necessary credentials to the user, see the following sections in the *Novell User Application: Administration Guide:*

- Removing Administrator Credential Restrictions
- Enabling the Test Page

Configuring IdentityIQ for Novell Identity Manager Integration

This section describes the procedure for Novell Identity Manager Integration.

Create Novell IDM application in IdentityIQ

Create a new IdentityIQ application of type Novell Identity Manager. Enter the following required Novell Identity Manager information:

- **Novell Version**: Version of the Novell Identity Manager. The version will accept a value of either 3 or 4. In case any other value is provided, the default value 3 is used.
- **User Application URL**: URL of the User Application, in the following format:
  http://applicationServerHost:port/applicationContext
- **Username**: User Application Administrator who has the rights to administer Web Services such as Provisioning using workflow tasks and Directory Abstraction Layer (VDX)
- **Password**: Password of the user specified above.

Define Provisioning Request Definition for Novell application

Enter the dn for the workflow created in step 5. on page 65 as application attributes:

1. To Create Vault User: **Create Vault User**
2. To Delete Vault User: **Delete Vault User**
3. To Disable Vault User: **Disable Vault User**
4. To Enable Vault User: **Enable Vault User**
5. To Modify Vault User: **Modify Vault User**
6. To all the operations on Resource Accounts on Applications (Entitlements). This includes Create, Delete, Disable, Enable Resource Accounts or Add/Remove Entitlements: Entitlements

   Note: Account Entitlement for any driver should contain “account” substring in it.

Run the Novell Application Creator Task

This task creates an IdentityIQ application for each connected systems driver in Novell® Identity Manager.

1. Select the Novell application created in the previous section.
2. Click **Fetch Novell Applications**. This shows all the connected systems. Select the ones with which you want IdentityIQ to integrate.
3. Click **Save and Execute**.
4. It will generate child applications for all drivers selected in IdentityIQ.

To fetch Account Status of Account Entitlement, add the following entry in Apllication's attribute, from the Application debug page:

<entry key="statusAttribute" value="loginDisabled"/>

In the above entry, **loginDisabled** represents the Status of Accounts on that driver.

Additional steps for Novell 4.0

Following manual configuration changes are required on IdentityIQ as there are number of configuration changes between Novell versions 3.x and 4.x:

1. After defining Novell application, change few attribute names in account and group schema for example, cN to CN, dirXML-Associations to DirXML-Associations. Check attribute names of Novell entity.

2. For generated child applications, add the following attributes in account schema:
   <AttributeDefinition multi="true" name="Account" type="string"/>

   <AttributeDefinition entitlement="true" managed="true" multi="true" name="Group" type="string"/>

   Set the **Group** attribute as group attribute of account schema.
3. Parent Novell application definition in IdentityIQ contains mapping between attribute names in version 4.0 which are used in application internally.
   ```
   <entry key="4.0">
       <value>
           <Map>
             <entry key="cN" value="CN"/>
             <entry key="dirXML-Associations" value="DirXML-Associations"/>
             <entry key="dirXML-EntitlementRef"
   value="DirXML-EntitlementRef"/>
             <entry key="LastName" value="LastName"/>
             <entry key="Description" value="Description"/>
             <entry key="objectClass" value="ObjectClass"/>
             <entry key="gUID" value="GUID"/>
             <entry key="loginDisabled" value="LoginDisabled"/>
             <entry key="dirXML-Driver" value="DirXML-Driver"/>
             <entry key="entitlementParameter">
                              <value>
                                     <String>{"ID":"nidm.com"}</String>
                              </value>
             </entry>
           </Map>
       </value>
   </entry>
   ```
4. By default, from IdentityIQ, while sending **User Account Entitlement** or **Group Entitlement**, parameter format is assumed as **Identity Manager 4**. If paramater format is configured as **legacy**, then add the following properties at child application for which parameter format is to be configured:
   ```
   <entry key="accountEntitlementParamFormat" value="legacy"/>
   <entry key="groupEntitlementParamFormat" value="legacy"/>
   ```
   If these properties are absent in application, then IdentityIQ will send parameter format in **Identity Manager 4** format.

For Novell version 4.0, **entitlementParameter** indicates name of domain name of Novell setup. Change the value from **nidm.com** to the domain name on which Novell is installed.

Aggregation from Novell PIM

To aggregate Novell users and resource accounts, create and execute an IdentityIQ Account Aggregation task. Include the Novell application in the applications to scan list.

Once aggregation is complete, all users in Novell have an account created that is associated with the Novell application and includes all of attributes defined in Novell application account schema. Additionally, all of the resource accounts are aggregated and associated with the child applications.

Note: **Before running the aggregation task, ensure** "Run the Novell Application Creator Task" on page 66 **has been executed.**

Chapter 7: SailPoint IdentityIQ IBM Tivoli Provisioning Integration Module

The following topics are discussed in this chapter:

Overview

This chapter is designed to provide the necessary procedures, configuration steps, and general product guidelines to successfully integrate IBM Tivoli® Identity Manager (ITIM) into your SailPoint IdentityIQ production environment.

This chapter is intended for ITIM and IdentityIQ System Administrators and assumes a high degree of technical knowledge of these systems.

Supported features

The IBM Tivoli Identity Manager Provisioning Integration Module provides the ability to provision Target Application accounts from IdentityIQ.

The Provisioning Integration Module supports the following functions:

- Account Aggregation on IBM Tivoli Identity Manager System
- Account Aggregation on Target Applications
- Create/Update/Delete Target Application Account
- Enable/Disable/Reset Password for Target Application Accounts

Supported platforms

- SailPoint IdentityIQ Deployment
- IBM Tivoli Identity Manager 4.6, 5.0, and 5.1
- IBM WebSphere Application Server 5.1, 6.1 and 7.0

General configuration

The installation steps for ITIM integrations vary based on the functions you wish to perform. IdentityIQ in conjunction with ITIM allows the following functionality:

- Aggregation
- Provisioning Entitlements in ITIM

Configuration for Aggregation

Aggregating from IBM Tivoli Identity Manager involves configuring the ITIM application settings within the IdentityIQ user interface.

ITIM has two types of objects that can be aggregated; people and accounts. IdentityIQ refers to these as identities and accounts (or links). To aggregate from ITIM, perform the following:

1. **Create An ITIM Application**: Create a new application using the IBM Tivoli Identity Manager connector and fill in the required parameters following the steps provided in the IdentityIQ User's Guide.
   Use the tenant DN search base. For example, **erglobalid=00000000000000000000,ou=example,dc=com**

   Leave the search filter blank. This is auto-generated correctly during aggregation. This application is used to aggregate ITIM person objects.
2. **Setup Correlation Attribute**: Create an identity attribute that is sourced from the **erglobalid** on the ITIM application and mark it as searchable. This is used to correlate ITIM accounts to this identity.
3. **Create ITIM Account Applications**: Run the ITIM Application Creator task to inspect ITIM and retrieve information about the ITIM services (applications). This task auto-generates an application for each service defined in ITIM.
4. **Setup Correlation on the ITIM Account Applications**: Set the correlation rule on the generated applications to Correlation - ITIM Account. This correlates the account to the identity using the **erglobalid**. If the rule is not listed by default, import it from the **$ITIM_INTEGRATION_PACKAGE/resources/ITIM-AccountCorrelationRule.xml** location.
5. **Aggregate**: Run aggregation for the ITIM application first and then for each ITIM account application.

Configuration for Provisioning

Provisioning entitlements and role assignments in ITIM requires the installation of IdentityIQ's ITIM integration web application in WebSphere with ITIM. This process varies slightly depending on the version of WebSphere.

IdentityIQ roles are queued and pushed in ITIM on a schedule. This is accomplished by using the Synchronize Roles task.

1. **Prepare the WAR:** The **iiqIntegration-ITIM.war** file contains a properties file named **itim.properties** with information about how to connect using ITIM. In order to execute, this must be edited to include appropriate information about the ITIM installation. Additionally, the **.war** file does not include any of the required **ITIM .jar** files since these can change depending on the version and fixpack level of ITIM. These need to be copied out of the ITIM lib directory and added to the **.war** file.
   a. Expand the **iiqIntegration-ITIM.war** file in a temporary directory.
   b. Edit the **WEB-INF/classes/itim.properties** file and change the properties match your environment.

Save the file with your changes. The following can be changed:

- **PLATFORM_URL: URL to use to communicate with ITIM.**
- **PLATFORM_PRINCIPAL: Principal to connect as.**
- **PLATFORM_CREDENTIALS: Password of the principal.**
- **SYSTEM_PRINCIPAL: Username for ITIM user that can perform operations**
- **SYSTEM_CREDENTIALS: Password for the system principal.**
- **TENANT_DN: The root DN of the ITIM tenant.**

c. Copy the required **ITIM .jar** files into the lib directory. These **.jar** files are located in the deployed ITIM ear directory.

- (*For ITIM 4.6*): Example ITIM ear directory: **$WAS_HOME/installedApps/itim/enRole.ear**

  Following are the required files:

  - **api_ejb.jar**
  - **itim_api.jar**
  - **itim_server.jar**
  - **jlog.jar**

- (For ITIM 5.0 and 5.1): Example ITIM ear directory:
  $WAS_HOME/profiles/<app server>/installedApps/<cell>/ITIM.ear
  Following are the required files:

  - **api_ejb.jar**
  - **itim_api.jar**
  - **itim_common.jar**
  - **itim_server_api.jar**
  - **jlog.jar**

d. Update the **iiqIntegration-ITIM.war** file to include the updated **itim.properties** and required **ITIM .jar** files.

For example,

```
jar uvf iiqIntegration-ITIM.war WEB-INF/classes/itim.properties \
WEB-INF/lib/api_ejb.jar WEB-INF/lib/itim_api.jar \
WEB-INF/lib/itim_common.jar WEB-INF/lib/itim_server_api.jar \
WEB-INF/lib/jlog.jar
```

2. Install the IdentityIQ **ITIM Integration Web Application**: In the WebSphere Administrative Console, navigate to Enterprise Applications and select **Install.**
   a. Select **iiqIntegration-ITIM.war** as the application to install and type **iiqitim** as the context root.
   b. Continue through the rest of the installation wizard accepting the defaults.
   c. When completed, click **Save** to save the changes to the master configuration.

3. **Setup the Integration Config**: The **IntegrationConfig** object holds information about how to connect IdentityIQ to ITIM and all of the configuration requirements for various functions. ITIM supports dual role push mode, which means that both detectable and assignable roles can be used. An example can be found in the ITIM integration folder within your IdentityIQ installation directory in the **$INSTALLDIR/integration/ITIM/server/ resources/exampleIntegration.xml** directory.

The main properties that need to be set are:

- **executor**: iiq.integration.itim.ITIMIntegrationExecutor
- **ApplicationRef**: The reference to the ITIM application
- **Attributes**=> **URL**: The URL to the IIQ web service on the ITIM server. For example, *https://myitim.example.com:9080/iiqitim/resources*
  Note: It is highly recommended that you use SSL when transmitting sensitive electronic information.
- **Attributes**=> **username**: ITIM user's credentials used for basic HTTP authentication.
- **Attributes**=>**password**: ITIM user's password used for basic HTTP authentication.
- **ManagedResources map**: Mappings of local IdentityIQ applications to ITIM services, including mappings of local IdentityIQ attribute names to ITIM service attribute names.

For more information, see Appendix: A: Common Identity Management Integration Configuration.

4. **Verify**: Be certain that the integration has been installed correctly by using the ping command in the integration console. If successful, this should respond and list version information about the ITIM jar files that were put into the **iiqIntegration-ITIM.war** file. Compare this version information against the version of the ITIM server to ensure correct operation.

Section 3: Service Integration Modules

This section contains the information on the following:

Chapter 8: Remedy Ticketing Service Integration Module

The following topics are discussed in this chapter:

Overview

Identity governance has become a strategic imperative for large, global enterprises and plays a critical role in enabling them to have visibility into the access privileges granted to their workers — and to answer the critical question: “Who has access to what? Without enterprise-wide visibility and control, serious gaps are left in the identity management process. These gaps can lead to failed audits, costly inefficiencies, and can place organizations at increased risk from entitlement creep, orphan accounts, Separation-of-Duty (SoD) violations, and the abuse of privileged user accounts. Many organizations have attempted to address identity governance requirements using automated provisioning systems like BMC User Administration Manager, and while these solutions help organizations automate the process of adding, modifying, and deleting user accounts across a set of target applications, they fall short of meeting the more stringent compliance demands and increasing security threats.

In order to deliver true identity governance, organizations must review and provide business-level oversight over all systems-at-risk, including many applications that are beyond the scope of provisioning. SailPoint IdentityIQ™, is an identity governance solution specifically designed to address these requirements and is integrated out-of-the-box with BMC Remedy Change Management to provide an end-to-end closed-loop process for Identity Governance and change request management.

The integration between the SailPoint and BMC Remedy solutions gives mutual customers a complementary identity governance and service management solution that works together to ensure strong controls are in place to meet ever stringent security and compliance requirements around user access to sensitive applications and data.

Supported features

- The Remedy ServiceDesk Integration Module supports creating ticket for all provisioning operations that can be performed on Target Application accounts.
- Remedy ServiceDesk Integration Module also support getting the status of the created tickets.

Supported platforms

IdentityIQ supports the following versions of BMC Remedy AR System:

- BMC Remedy AR System 7.1.00
- BMC Remedy AR System 8.0.00
- BMC Remedy AR System 8.1.00

Pre-requisites

- BMC Remedy Change Management Application must be installed.
- Ensure that the following softwares are operating correctly:
    - BMC Remedy AR System
    - BMC Remedy Change Management Application

Basic configuration

The integrated solution speeds the detection and remediation of identity management issues that increase the risk of compliance violations or security breaches, such as orphaned accounts, policy violations, and inappropriate access privileges. Organizations can take advantage of a centralized approach spanning thousands of users and hundreds of resources to strengthen IT controls and provide proof of compliance to auditors and executive management. The seamless integration of SailPoint and BMC Remedy eliminates the need to build and maintain a custom integration, and speeds time-to-deployment.

For any IT resources managed by BMC Remedy Service Desk, IdentityIQ automatically creates a trouble ticket within Remedy Service Desk, passing along all relevant identity data and reviewer comments to populate the ticket.

To ensure revocation requests get delivered and implemented, IdentityIQ manages all remediation and revocation requests within a guaranteed delivery model.

To determine the status of user accounts, IdentityIQ performs closed-loop audits on remediation requests and compares the actual state of user privileges with the original change request. If the request is still open, an alert will be sent to the reviewer for prompt action and closure.

The integration itself has been designed to be quick to install and easy to use. It makes use of Web Services via the Remedy Mid Tier to broker communications between the SailPoint server and the AR System server. On the backside of a user recertification, policy remediation action or access request action, the IdentityIQ server will direct provisioning and service desk requests to the configured implementers. Based on the IntegrationConfig configured for each target application, service desk request are issues to a given remediation/implementation point. Once the IntegrationConfig for Remedy has been loaded into the IdentityIQ server, all change/remediation actions result in the creation of new service desk request.

At the completion of the change control cycle within IdentityIQ, an “Open Ticket” request is made over the appropriate SOAP channel to the Mid Tier. From here change request tickets are opened and the new ticket number is returned to IdentityIQ. The schema for the service request is defined in the IntegrationConfig and allows for the flexibility to transfer complete details on the service desk request. The default settings will create a basic ticket as shown in the following figure (Figure 2—Change request).

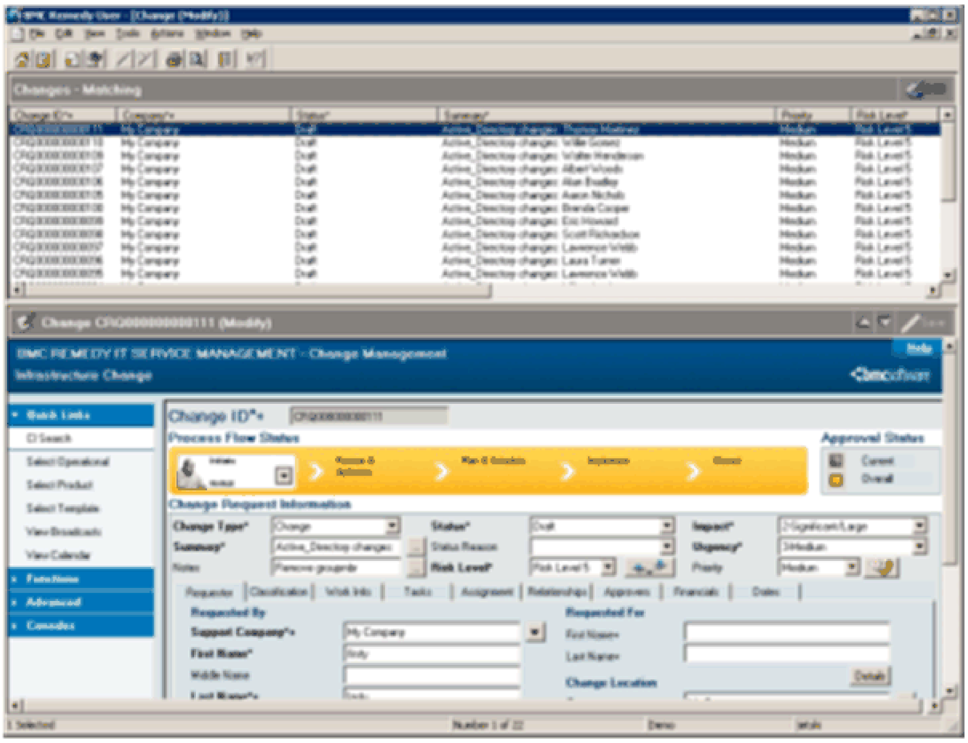


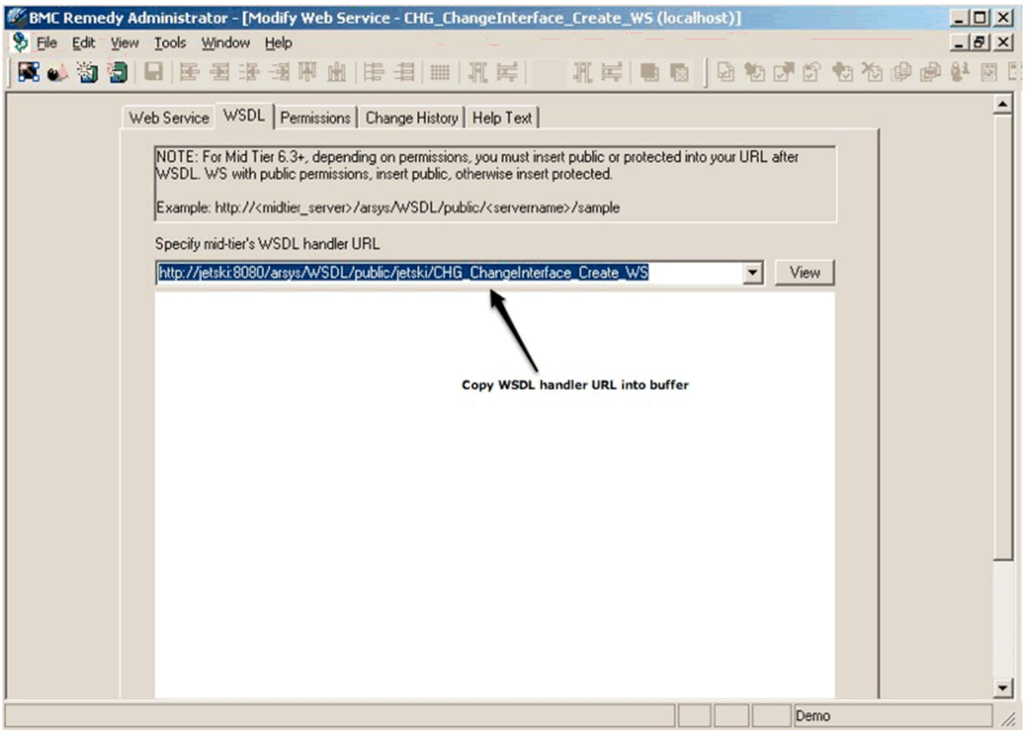
Figure 2—Change request

Configuring BMC Remedy AR System for IdentityIQ Integration

This section provides the required information for configuring IdentityIQ to integrate with BMC Remedy Action Request System (AR System). This integration enables IdentityIQ to create Change Management tickets for requested revocations, track ticket numbers in association with revocation tasks, and update IdentityIQ with the status of current Change Management tickets.

The following steps should be performed to modify the default Remedy integration configuration for a specific BMC Remedy application instance.

1. Confirm the default Remedy Change Management Application Web Services exist. This is done by launching the BMC Remedy Administrator, expanding the appropriate server object and clicking on the “Web Services” object.
2. Next, obtain the environment-specific Web Service “endpoint” by performing the following steps:
    a. Double-click on the Web Service and select the WSDL tab. Copy the WSDL handler URL into your buffer (For example, Ctrl-C)


BMC Remedy Administrator - [Modify Web Service - CHG_ChangeInterface_Create_WS (localhost)]

NOTE: For Mid Tier 6.3+, depending on permissions, you must insert public or protected into your URL after WSDL. WS with public permissions, insert public, otherwise insert protected.

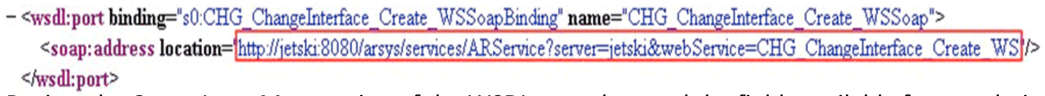
Example: http://<midtier_server>/arsys/WSDL/public/<servername>/sample

Specify mid-tier's WSDL handler URL

http://jetski:8080/arsys/WSDL/public/jetski/CHG_ChangeInterface_Create_WS

b. With a web browser, visit the WSDL URL for the web service by entering the URL into the browser address field and pressing return.

c. Search for **soap:address location=** to find the endpoint URL. Copy this value. It will be used to replace the endpoint URL in the default IdentityIQ Remedy IntegrationConfig object.


```
- <wsdl:port binding="s0:CHG_ChangeInterface_Create_WSSoapBinding" name="CHG_ChangeInterface_Create_WSSoap">
    <soap:address location="http://jetski:8080/arsys/services/ARService?server=jetski&webService=CHG_ChangeInterface_Create_WS" />
  </wsdl:port>
```

d. Review the CreateInputMap section of the WSDL to understand the fields available for population through the Web Service. These fields should correspond to the fields listed in the <soapenv:Body> section of the default IdentityIQ IntegrationConfig object

3. Once you are familiar with the WSDL, modify the default IdentityIQ Remedy integration using the information collected about the web service.

   a. In the <IntegrationConfig> element of the integration configuration, modify the **username** and **password** entries in the attributes map to contain the credentials required for authentication to the web service.

   b. In the <IntegrationConfig> element of the integration configuration, modify the provision entry of the Attributes map by setting the endpoint, and, if necessary, the namespace, the prefix, the responseElement, and the soapMessage attributes (the default values: IdentityIQ Remedy IntegrationConfig):

      i. Set the value for endpoint to the value located in the WSDL earlier.

Note: The value in the IdentityIQ integration configuration must be a valid HTTP URL and have any special characters escaped. The most common change that must be made is to replace all & symbols with &amp;

      ii. The value for namespace comes from the **targetNamespace** attribute of the **xsd:schema** element in the WSDL.

iii. The value for prefix is the prefix of the XML elements that will be contained in the SOAP response sent by the mid tier server.

iv. The value for responseElement should be the ARS form field that corresponds to the id of the form that the web service creates.

v. The value for soapMessage should be the SOAP message body that IdentityIQ will send to ARS. The exact format of this message is a function of the form that is published as described by the form's WSDL. The XML elements in the **soapenv:Body** element should be changed to match the ARS form fields for the published web service. Each required ARS form field must have an element in the SOAP message. The value can be fixed or can be a variable that will be substituted using IdentityIQ's Velocity templating

The information in the reference section above show the variables that are provided and the example integration configuration provides examples of how they are used.

Configuring IdentityIQ for BMC Remedy Action Request System Integration

This is intended as an introduction to the configuration needed to integrate IdentityIQ with the BMC Remedy Action Request System. This integration enables IdentityIQ to interact with many of the product solutions that are built on top of the AR System Server including BMC Remedy Change Management, BMC Remedy IT Service Management Suite, and BMC Remedy Service Desk.

SailPoint provides a default Remedy integration configuration. This configuration implements the stock integration between IdentityIQ and the Remedy Change Management Application to fulfill creation of tickets based on IdentityIQ access certification remediation events.

The default configuration is located in ***iiqHome*/WEB-INF/config/remedy-Integration.xml** directory, where *iiqHome* is the location where IdentityIQ was installed.

For BMC Remedy AR System version 8.0 and 8.1, an additional sample xml file is located in ***iiqHome*/WEB-INF/config/remedyV8-integration.xml** directory.

This section explains the various entries that are specific for this integration. For more information of the entires in the **IntegrationConfig** file, see Appendix A: Common Identity Management Integration Configuration.

The integration configuration must include the following entries:

- **endpoint**: URL to the web service
- **namespace**: namespace of the XML returned by the web service
- **prefix**: prefix associated with the namespace

The integration configuration includes the following entries if the web service side of the integration is configured for authentication using the SOAP authentication specifications:

- username
- password
- authentication
- locale
- timeZone
- statusMap

The integration configuration includes the following entries if the http authentication is configured:

- basicAuthType: if http authentication is configured the value of basicAuthType is true.
- httpUserName
- httpUserPass

The web services and authentication entries are consumed by configuration entries for each web service. They can be positioned either within the configuration entries themselves or as children of the Attributes element. Entries that are children of the Attributes element can be thought of as global values, while entries within the configuration entities can be thought of as local.

For example, if both entries share the same authentication credentials, those credentials might be placed in the Attributes element as peers of the configuration entries and the integration code searches the parent entry for the credentials if they are not found in the configuration entries. Conversely, if the configuration entries have different endpoints (are handled by separate web services), each configuration entry specifies the endpoint of the web service to call and any value outside of the configuration entry is ignored.

There are two supported configuration entries for integration with Remedy. These entries are children of the integration Attributes element:

- getRequestStatus
- provison

The values of each are Map elements containing key/value pairings of the configuration data. They contain the specific data needed by the getRequestStatus()and provision() methods of the IdentityIQ integration executor and correspond to Remedy Web Service methods.

The **getRequestStatus** and **provison** entries contain the following entries:

- **soapMessage** (required): full XML template of the entire SOAP envelope that is sent to the web service. The integration code first runs this template through Apache's Velocity template engine to provide the data needed by the web service.
- **responseElement** (required): name of the element containing the results of the web service call (for example, the element containing the ticket number opened by the web service in response to the call from IdentityIQ).
- **statusMap** (optional, see “Sample getRequestStatus entry” on page 81 for an example)
- **username** (optional)
- **password** (optional)
- **authentication** (optional)
- **locale** (optional)
- **timeZone** (optional)
- **endpoint** (optional)
- **namespace** (optional)
- **prefix** (optional)

Before a template is sent to the web service, it is processed by the **Velocity template engine**. The integration code provides different data objects to Velocity for evaluation based on the integration method.

The **provision** call passes the following objects to Velocity:

- **config**: the integration configuration for provision, represented as a Map
- **provisioningPlan**: the data model of the provision request

The **getRequestStatus** call passes the following objects to Velocity:

- **config**: the integration configuration for getRequestStatus, represented as a Map
- **requestID**: the string ID of the request whose status is being queried

Both calls have access to a timestamp variable containing a current Date object and a dateFormatter object. The dateFormatter is built using an optional dateFormat attribute from the **config** object. If the dateFormat attribute does not exist, the formatter defaults to the pattern EEE, d MMM yyyy HH:mm:ss z.

Sample getRequestStatus entry

Note: The entries contained in the Map are the only required entries. Any authentication information required by this integration is inherited from the parent Attributes element.

```
<entry key="getRequestStatus">
 <value>
  <Map>
    <entry key="responseElement" value="Status"/>
    <entry key="soapMessage">
      <!-- XML template – DO NOT add line breaks before the CDATA! -->
      <value><String><![CDATA[<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
#if ($config.username)
<soapenv:Header>
<ns1:AuthenticationInfo xmlns:ns1="urn:AuthenticationInfo">
         <ns1:userName>$config.username</ns1:userName>
         <ns1:password>$config.password</ns1:password>
    #if ($config.authentication)
         <ns1:authorization>$config.authentication</ns1:password>
#end
#if ($config.locale)
         <ns1:locale>$config.locale</ns1:password>
#end
#if ($config.timeZone)
         <ns1:timeZone>$config.timeZone</ns1:password>
#end
</ns1:AuthenticationInfo>
</soapenv:Header>
#end
<soapenv:Body>
     <iiq:Get xmlns:iiq="urn:GetAgreementWebService">
       <iiq:Issue_ID>$requestID</iiq:Issue_ID>
     </iiq:Get>
   </soapenv:Body>
</soapenv:Envelope>
]]>
          </value>
        </entry>
      </Map>
    </value>
</entry>
```

Sample provision entry

Note: This Map contains its own web services information. Any authentication information required by this integration is inherited from the parent Attributes element.

```
<entry key="provision">
   <value>
     <Map>
         <entry key="endpoint" value="http://my.server.com:8080/path/to/WS"/>
         <entry key="namespace" value="urn:openTicketWebService"/>
         <entry key="prefix" value="xyz"/>
         <entry key="responseElement" value="Issue_ID"/>
         <entry key="soapMessage">
            <!-- XML template – DO NOT add line breaks before the CDATA! -->
            <value><String><![CDATA[<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
#if ($config.username)
<soapenv:Header>
<ns1:AuthenticationInfo xmlns:ns1="urn:AuthenticationInfo">
         <ns1:userName>$config.username</ns1:userName>
         <ns1:password>$config.password</ns1:password>
#if ($config.authentication)
         <ns1:authorization>$config.authentication</ns1:password>
#end
#if ($config.locale)
         <ns1:locale>$config.locale</ns1:password>
#end
#if ($config.timeZone)
         <ns1:timeZone>$config.timeZone</ns1:password>
#end
</ns1:AuthenticationInfo>
</soapenv:Header>
#end
<soapenv:Body>
   <iiq:Get xmlns:iiq="urn:openTicketWebService">
     <iiq:Submitter>
     #foreach ($req in $provisionPlan.requesters)
        $req.name
     #end
     </iiq:Submitter>
     </iiq:SubmitDate>$timestamp</iiq:SubmitDate>
     <iiq:Summary>
        Remediation request from IIQ
     </iiq:Summary>
     <iiq:Description>
        Remove Active Directory for $provisionPlan.identity.fullname
     </iiq:Description>
     <iiq:Issue_ID>$requestID</iiq:Issue_ID>
    </iiq:Get>
   </soapenv:Body>
</soapenv:Envelope>
]]>
```

```
            </value>
          </entry>
        </Map>
      </value>
    </entry>
```

Sample statusMap entry

The **noMappingFromWS** entries are placeholders as there are no results from the web service corresponding to those IdentityIQ result codes.

```
<entry key="statusMap">
    <value>
      <Map>
        <entry key="Closed" value="success" />
        <entry key="Rejected" value="failure" />
        <entry key="Draft" value="inProcess" />
        <entry key="Pending" value="inProcess" />
        <entry key="noMappingFromWS" value="retry" />
        <entry key="noMappingFromWS" value="warning" />
      </Map>
    </value>
</entry>
```

Sample scenario

The sample integration scenario is built around a sample system. In the sample scenario SailPoint IdentityIQ (IIQ) will be issuing a change request to BMC Remedy Change Management (RCM) based on the results of a scheduled user entitlement and access review. As a result of remediation actions in this account recertification process, IdentityIQ will open change requests to control the flow of the manual remediation process.

Scenario

1. The ComplianceManager1 schedules an access review for a business critical application:
    a. The certification is scheduled and assigned to ApplicationOwner1.
    b. ApplicationOwner1 receives an email with a link to the online certification process as scheduled. The link is followed to the open certification.
    c. ApplicationOwner1 decides that GroupA on system LDAP should be removed.
    d. ApplicationOwner1 decides that RoleA on system RDBMS should be removed.
    e. ApplicationOwner1 completes the certification and signs off the process.
2. IdentityIQ evaluates the provisioning plan to enact the remediation requests from the certification:
    a. IdentityIQ policy describes the integration execution path for LDAP as being via an automated provisioning system.
    b. IdentityIQ policy describes the integration execution path for RDBMS as being via an automated RCM integration.
3. IdentityIQ creates a service request in RCM:
    a. IdentityIQ uses the **provision** interface to open a service request within ServiceNow, passing in details of the changes required to the RDBMS system.
    b. RCM responds with the service request number.
    c. IdentityIQ stores the service request number for later audit and review.

Known/Open Issues

Following is the known/open issue of Remedy Ticketing Service Integration Module:

- Only one ticket will be created for all account requests in the provisioning plan

Chapter 9: SailPoint IdentityIQ Integration for ServiceNow Service Integration Module

The following topics are discussed in this chapter:

Overview

Identity governance has become a strategic imperative for large, global enterprises and plays a critical role in enabling them to have visibility into the access privileges granted to their workers — and to answer the critical question: "Who has access to what? Without enterprise-wide visibility and control, serious gaps are left in the identity management process. These gaps can lead to failed audits, costly inefficiencies, and can place organizations at increased risk from entitlement creep, orphan accounts, Separation-of-Duty (SoD) violations, and the abuse of privileged user accounts. Many organizations have attempted to address identity governance requirements using automated provisioning systems like IdentityIQ, and while these solutions help organizations automate the process of adding, modifying, and deleting user accounts across a set of target applications, they fall short of meeting the more stringent compliance demands and increasing security threats.

In order to deliver true identity governance, organizations must review and provide business-level oversight over all systems-at-risk, including many applications that are beyond the scope of provisioning. SailPoint IdentityIQ™, is an identity governance solution specifically designed to address these requirements and is integrated out-of-the-box with ServiceNow Service Desk to provide an end-to-end closed-loop process for Identity Governance and change request management.

The integration between the SailPoint and ServiceNow Service Desk gives mutual customers a complementary identity governance and service management solution that works together to ensure strong controls are in place to meet ever stringent security and compliance requirements around user access to sensitive applications and data.

Supported features

- The ServiceNow ServiceDesk Integration Module supports creating ticket for all provisioning operations that can be performed on Target Application accounts.
- ServiceNow ServiceDesk Integration Module also support getting the status of the created tickets.

Pre-requisites

ServiceNow Instance should be up and running.

Basic configuration

The integrated solution speeds the detection and remediation of identity management issues that increase the risk of compliance violations or security breaches, such as orphaned accounts, policy violations, and inappropriate access privileges. Organizations can take advantage of a centralized approach spanning thousands of users and hundreds of resources to strengthen IT controls and provide proof of compliance to auditors and executive management. The seamless integration of SailPoint and ServiceNow eliminates the need to build and maintain a custom integration, and speeds time-to-deployment.

For any IT resources managed by ServiceNow Service Desk, IdentityIQ automatically creates a trouble ticket within ServiceNow Service Desk, passing along all relevant identity data and reviewer comments to populate the ticket.

To ensure revocation requests get delivered and implemented, IdentityIQ manages all remediation and revocation requests within a guaranteed delivery model.

To determine the status of user accounts, IdentityIQ performs closed-loop audits on remediation requests and compares the actual state of user privileges with the original change request. If the request is still open, an alert will be sent to the reviewer for prompt action and closure.

The integration itself has been designed to be quick to install and easy to use. It makes use of Web Services for communications between the SailPoint server and the ServiceNow. On the backside of a user recertification, policy remediation action or access request action, the IdentityIQ server will direct provisioning and service desk requests to the configured implementers. Based on the IntegrationConfig configured for each target application, service desk request are issues to a given remediation/implementation point. Once the IntegrationConfig for ServiceNow has been loaded into the IdentityIQ server, all change/remediation actions result in the creation of new service desk request as shown in .

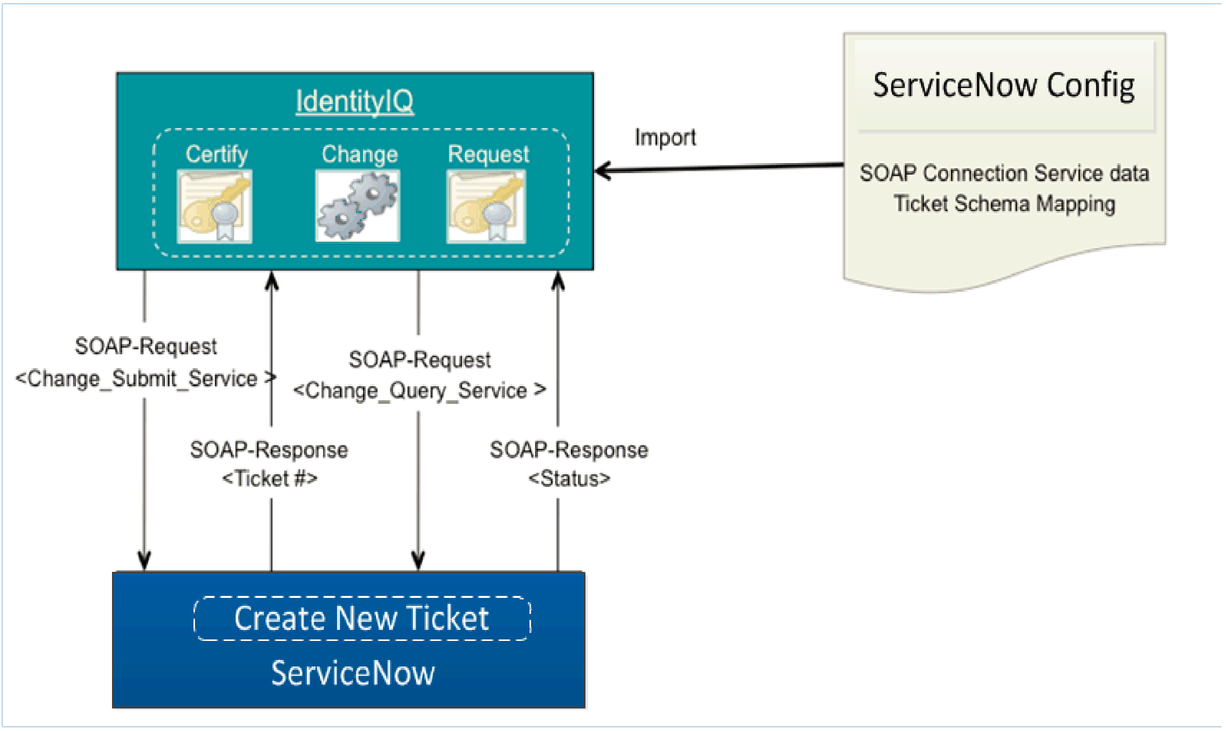
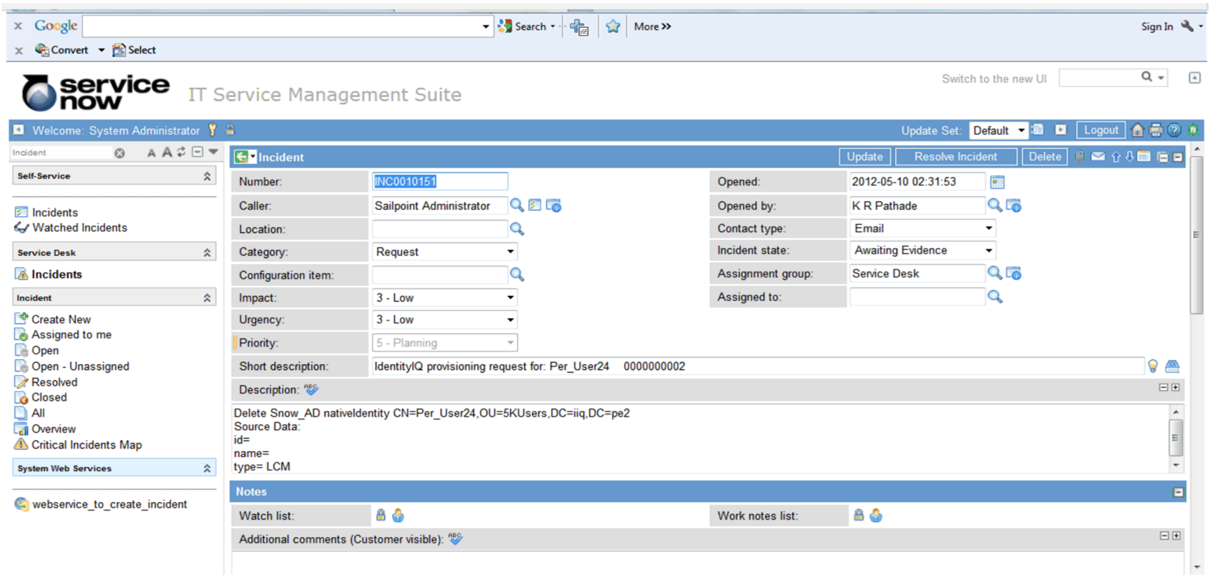


Figure 3—Basic configuration

At the completion of the change control cycle within IdentityIQ, an "Open Ticket" request is made over the appropriate SOAP channel to the ServiceNow web service. From here, tickets are opened and the new ticket number is returned to IdentityIQ. The schema for the service request is defined in the IntegrationConfig and allows for the flexibility to transfer complete details on the service desk request. The default settings will create a basic ticket as shown in .



Number:	INC0010151	Opened:	2012-05-10 02:31:53
Caller:	Sailpoint Administrator	Opened by:	K R Pathade
Location:		Contact type:	Email
Category:	Request	Incident state:	Awaiting Evidence
Configuration item:		Assignment group:	Service Desk
Impact:	3 - Low	Assigned to:	
Urgency:	3 - Low		
Priority:	5 - Planning		

Short description: IdentityIQ provisioning request for: Per_User24 0000000002

Description:

```
Delete Snow_AD nativeIdentity CN=Per_User24,OU=5KUsers,DC=iiq,DC=pe2
Source Data:
id=
name=
type= LCM
```

Watch list:

Work notes list:

Additional comments (Customer visible):

Figure 4—Service Desk request

Configuring ServiceNow for IdentityIQ Integration

This section provides the required information for configuring IdentityIQ to integrate with ServiceNow. This integration enables IdentityIQ to create tickets for requested revocations, track ticket numbers in association with revocation tasks, and update IdentityIQ with the status of current tickets.

Before you begin

Ensure that ServiceNow instance is operating correctly.

Prerequisities:

- When running a ServiceNow integration on an application server usin Java 6, the JRE must be directed to use the SOAP/SAAJ implementation provided by Axis2. To do this, add the following Java options to the application server:
  - -Djavax.xml.soap.SOAPConnectionFactory=org.apache.axis2.saaj.SOAPConnectionFactoryImpl
  - -Djavax.xml.soap.MessageFactory=org.apache.axis2.saaj.MessageFactoryImpl
  - -Djavax.xml.soap.SOAPFactory=org.apache.axis2.saaj.SOAPFactoryImpl

  Without these options, a ClassCastException error or Signature creation failed error will occur when the ServiceNow integration is used.
- **Ensure that the Service Now is up and running**: To ensure that Service Now is up and web services component is running, access the following WSDL:

  https://<ServiceNowHost>/incident.do?WSDL

  Alternatively, use a Soap UI tool to submit a simple request (for example, incident). For convenience you can use the basic authentication mechanism for authorization with SOAP UI tool to confirm that the web service layer is functional.
- **Ensure that ServiceNowIntegrationExecutor is being called**: ServiceNowIntegrationExecutor class is responsible for creating and sending SOAP requests to Service Now. You can add a simple SystemOut statement in **ServiceNowIntegrationRule** to ensure that this is being called when a provisioning request is submitted for this integration.

ServiceNow Integration

This is intended as an introduction to the configuration needed to integrate IdentityIQ with the ServiceNow. This integration enables IdentityIQ to interact with ServiceNow Service Desk.

SailPoint provides a default ServiceNow configuration. This configuration implements the stock integration between IdentityIQ and the ServiceNow to fulfill creation of tickets based on IdentityIQ access certification remediation events.

The default configuration is located in ***iiqHome*/WEB-INF/config/sampleServiceNowintegration.xml** directory, where *iiqHome* is the location where IdentityIQ was installed.

This section explains the various entries that are specific for this integration. For more information of the entires in the **IntegrationConfig** file, see Appendix A: Common Identity Management Integration Configuration.

The integration configuration must include the following entries:

- **endpoint**: URL to the web service
- **namespace**: namespace of the XML returned by the web service
- **prefix**: prefix associated with the namespace

The integration configuration includes the following entries if the web service side of the integration is configured for authentication using the SOAP authentication specifications:

- username (Can be blank if **Require basic authorization for incoming SOAP requests** is not selected on ServiceNow side)
- password (Can be blank if **Require basic authorization for incoming SOAP requests** is not selected on ServiceNow side)
- authentication
- locale
- timeZone
- statusMap

The integration configuration also includes the following properties if WS-Security is enabled on service-now side:

- authType
- keystorePath
- keystorePass
- keystoreType
- alias
- keyPass

For more information on enabling the WS-Security on ServiceNow side, see “Configuration required on ServiceNow side for WS-Security” on page 94.

The web services and authentication entries are consumed by configuration entries for each web service. They can be positioned either within the configuration entries themselves or as children of the Attributes element. Entries that are children of the Attributes element can be thought of as global values, while entries within the configuration entities can be thought of as local.

For example, if both entries share the same authentication credentials, those credentials might be placed in the Attributes element as peers of the configuration entries and the integration code searches the parent entry for the credentials if they are not found in the configuration entries. Conversely, if the configuration entries have different endpoints (are handled by separate web services), each configuration entry specifies the endpoint of the web service to call and any value outside of the configuration entry is ignored.

There are two supported configuration entries for integration with ServiceNow. These entries are children of the integration Attributes element:

- getRequestStatus
- provison

The values of each are Map elements containing key/value pairings of the configuration data. They contain the specific data needed by the getRequestStatus()and provision() methods of the IdentityIQ integration executor and correspond to ServiceNow Web Service methods.

The **getRequestStatus** and **provison** entries contain the following entries:

- **soapMessage** (required): full XML template of the entire SOAP envelope that is sent to the web service. The integration code first runs this template through Apache's Velocity template engine to provide the data needed by the web service.

- **responseElement** (required): name of the element containing the results of the web service call (for example, the element containing the ticket number opened by the web service in response to the call from IdentityIQ).
- **statusMap** (optional, see "Sample getRequestStatus entry" on page 90 for an example)
- **username** (Can be blank if **Require basic authorization for incoming SOAP requests** is not selected on ServiceNow side)
- **password** (Can be blank if **Require basic authorization for incoming SOAP requests** is not selected on ServiceNow side)
- **authentication**
- **locale** (optional)
- **timeZone** (optional)
- **endpoint** (optional)
- **namespace** (optional)
- **prefix** (optional)
- **authType** (optional): Use "WS-Security" if WS Security is enabled on service-now side. Otherwise leave it blank.
- **keystorePath** (optional): Full path of keystore
- **keystorePass** (optional): Password of keystore
- **keystoreType** (optional): Type of keystore. For example, jks
- **alias** (optional): The alias of certificate in keystore
- **keyPass** (optional): The password of alias

Before a template is sent to the web service, it is processed by the **Velocity template engine**. The integration code provides different data objects to Velocity for evaluation based on the integration method.

The **provision** call passes the following objects to Velocity:

- **config**: the integration configuration for provision, represented as a Map
- **provisioningPlan**: the data model of the provision request

The **getRequestStatus** call passes the following objects to Velocity:

- **config**: the integration configuration for getRequestStatus, represented as a Map
- **requestID**: the string ID of the request whose status is being queried

Both calls have access to a timestamp variable containing a current Date object and a dateFormatter object. The dateFormatter is built using an optional dateFormat attribute from the **config** object. If the dateFormat attribute does not exist, the formatter defaults to the pattern EEE, d MMM yyyy HH:mm:ss z.

Sample getRequestStatus entry

Note: The entries contained in the Map are the only required entries. Any authentication information required by this integration is inherited from the parent Attributes element.

```
 <entry key="getRequestStatus">
     <value>
       <Map>
         <entry key="endpoint" value="https://demo.service-now.com/incident.do?SOAP"/> <!-- e.g.
https://demo.service-now.com/incident.do?SOAP !-->
  <entry key="prefix" value="q0"/>
         <entry key="responseElement" value="incident_state"/>
  <entry key="closureInfoResponseElement" value="close_code"/>
```

```
        <entry key="soapMessage">
          <!-- XML template -->
          <value>
            <String><![CDATA[<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:q0="http://www.service-now.com/incident" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <q0:getRecords>
      <number>$requestID</number>
    </q0:getRecords>
  </soapenv:Body>
</soapenv:Envelope>
]]>
                    </String>
                  </value>
                </entry>
              </Map>
            </value>
          </entry>
        </Map>
    </Attributes>
```

Sample provision entry

Note: This Map contains its own web services information. Any authentication information required by this integration is inherited from the parent Attributes element.

```
<entry key="provision">
          <value>
            <Map>
        <entry key="endpoint" value="https://demo.service-now.com/incident.do?SOAP"/> <!-- e.g.
https://demo.service-now.com/incident.do?SOAP !-->
<entry key="prefix" value="q0"/>
        <entry key="responseElement" value="number"/>
        <entry key="soapMessage">
          <!-- XML template -->
              <value>
                <String><![CDATA[<?xml version="1.0" encoding="UTF-8"?>

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:q0="http://www.service-now.com/incident" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body>
<q0:insert>
  <assignment_group>Service Desk</assignment_group>
  <category>request</category>
  <contact_type>email</contact_type>
  <impact>3</impact>
  <incident_state>1</incident_state>
<caller_id>$provisioningPlan.integrationData.requester</caller_id>
  <opened_by>$config.username</opened_by>
  <priority>5</priority>
  <short_description>IdentityIQ provisioning request for: $provisioningPlan.integrationData.identityName
$!provisioningPlan.integrationData.identityRequestId
 </short_description>
  <description>#foreach ($request in $provisioningPlan.accountRequests)
#if ($request.attributeRequests)
#foreach ($att in $request.attributeRequests)
$request.application $att.operation $att.name $att.value
#end
#else
$request.operation $request.application nativeIdentity $request.nativeIdentity
#end
#end
Source Data:
id= $!provisioningPlan.integrationData.sourceId
name= $!provisioningPlan.integrationData.sourceName
type= $!provisioningPlan.integrationData.sourceType</description>
  <urgency>3</urgency>
</q0:insert>
</soapenv:Body>
</soapenv:Envelope>
 ]]>
           </String>
      </value>
    </entry>
  </Map>
</value>
  </entry>
```

Sample statusMap entry

The **noMappingFromWS** entries are placeholders as there are no results from the web service corresponding to those IdentityIQ result codes.

```
 <entry key="statusMap">
      <value>
        <Map>
          <entry key="1" value="queued" />
<entry key="2" value="queued" />
<entry key="3" value="queued" />
<entry key="4" value="queued" />
<entry key="5" value="queued" />
<entry key="6" value="committed" />
          <entry key="7" value="committed" />
        </Map>
      </value>
    </entry>

<entry key="statusMapCloserCode">
      <value>
        <Map>
          <entry key="Solved (Work Around)" value="committed" />
<entry key="Solved (Permanently)" value="committed" />
<entry key="Solved Remotely (Work Around)" value="committed" />
```

```
<entry key="Solved Remotely (Permanently)" value="committed" />
<entry key="Closed/Resolved by Caller" value="committed" />
<entry key="Not Solved (Not Reproducible)" value="failed" />
          <entry key="Not Solved (Too Costly)" value="failed" />
        </Map>
      </value>
    </entry>
```

Configuration procedure

The following steps should be performed to modify the default ServiceNow integration configuration for a specific ServiceNow instance.

1. Obtain the environment-specific Web Service "endpoint", for example, https://demo.service-now.com/incident.do?SOAP
   Either a web service can be created a web service pointing to system table can be used, for example, https://demo.service-now.com/incident.do?SOAP
2. Once you are familiar with the WSDL, modify the default IdentityIQ ServiceNow configuration using the information collected about the web service.
   a. In the <IntegrationConfig> element of the integration configuration, modify the **username** and **password** entries in the attributes map to contain the credentials required for authentication to the web service.
   b. If you have enabled WS-Security on ServiceNow side, modify entries for **authType**, **keystorePath**, **keystorePass**, **keystoreType**, **alias**, **keyPass** to contain keystore related details.
   c. In the <IntegrationConfig> element of the integration configuration, modify the provision entry of the Attributes map by setting the endpoint, and, if necessary, the namespace, the prefix, the responseElement, and the soapMessage attributes (the default values: IdentityIQ ServiceNow IntegrationConfig):

      i. Set the value for endpoint to the value located in the WSDL earlier.

      Note: The value in the IdentityIQ integration configuration must be a valid HTTP URL and have any special characters escaped. The most common change that must be made is to replace all & symbols with &amp;

      ii. The value for namespace comes from the **targetNamespace** attribute of the **xsd:schema** element in the WSDL.

      iii. The value for prefix is the prefix of the XML elements that will be contained in the SOAP response.

      iv. The value for responseElement should be the ServiceNow form field that corresponds to the id of the form that the web service creates.

      v. The value for soapMessage should be the SOAP message body that IdentityIQ will send to ServiceNow. The exact format of this message is a function of the form that is published as described by the form's WSDL. The XML elements in the **soapenv:Body** element should be changed to match the ServiceNow form fields for the published web service. Each required ServiceNow form field must have an element in the SOAP message. The value can be fixed or can be a variable that will be substituted using IdentityIQ's Velocity templating

The information in the reference section above show the variables that are provided and the example integration configuration provides examples of how they are used.

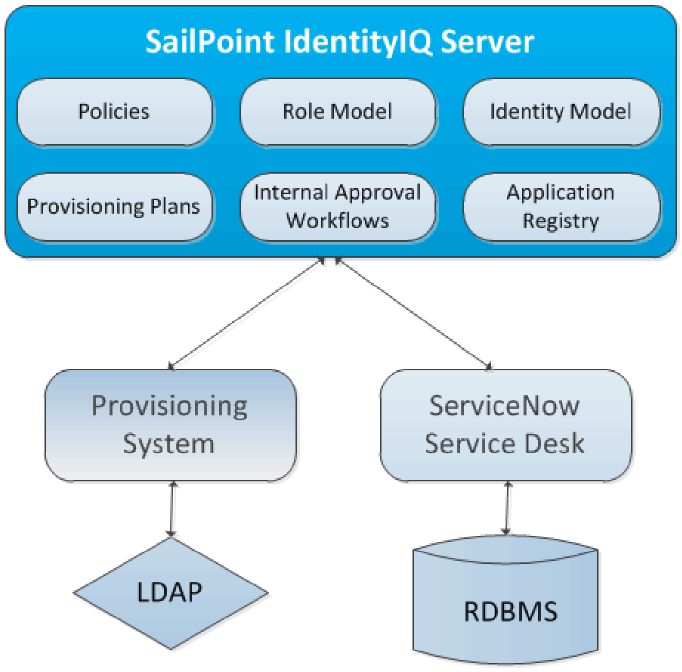
Configuration required on ServiceNow side for WS-Security

Perform the following steps to enable WS-Security on ServiceNow side:

1. Login to service-now instance with user having access to do system changes for example, admin.
2. Navigate to **System Definition => Certificates** and click on **New** button.
3. Enter some name for the certificate.
4. In command prompt, navigate to directory where the certificate is located.
5. Enter the following command, where **mycert.pem** is the name of the certificate:
   type mycert.pem
   The following similar output is displayed:
   ```
   -----BEGIN CERTIFICATE-----
   MIIDOTCCAiECBE/6tzswDQYJKoZIhvcNAQEEBQAwYTELMAkGA1UEBhMCSU4xCzAJBgNVBAgTAk1I
   ...
   ...
   -----END CERTIFICATE-----
   ```
6. Copy all above content (including Begin and End Certificate lines) and navigate to service-now and paste the above contents into **PEM Certificate** field.
7. Save the certificate.
8. Navigate to System **Web Services =>WS Security Profiles** and click on **New**.
9. In **Type** field, select X509 and iIn **X509 Certificate** field, select certificate which we uploaded.
10. Select bind session checkbox.
11. In Run as user field, select name of user on behalf of whom, you want to execute web-services for example, System Administrator.
12. Click on **Submit** button.
13. Navigate to **System Web Services => Properties**.
14. Select the **Require WS-Security header verification for all incoming SOAP requests** check box and save it.

Sample scenario

The sample integration scenario is built around a sample system as shown in the following figure:



In the sample scenario SailPoint IdentityIQ (IIQ) will be issuing change request to ServiceNow based on the results of a scheduled user entitlement and access review. As a result of remediation actions in this account recertification process, IdentityIQ will open incident to control the flow of the manual remediation process.

Scenario

1. The ComplianceManager1 schedules an access review for a business critical application:
    a. The certification is scheduled and assigned to ApplicationOwner1.
    b. ApplicationOwner1 receives an email with a link to the online certification process as scheduled. The link is followed to the open certification.
    c. ApplicationOwner1 decides that GroupA on system LDAP should be removed.
    d. ApplicationOwner1 decides that RoleA on system RDBMS should be removed.
    e. ApplicationOwner1 completes the certification and signs off the process.
2. IdentityIQ evaluates the provisioning plan to enact the remediation requests from the certification:
    a. IdentityIQ policy describes the integration execution path for LDAP as being via an automated provisioning system.
    b. IdentityIQ policy describes the integration execution path for RDBMS as being via an automated ServiceNow integration.
3. IdentityIQ creates a service request in ServiceNow:
    a. IdentityIQ uses the **provision** interface to open a service request within ServiceNow, passing in details of the changes required to the RDBMS system.
    b. ServiceNow responds with the service request number.
    c. IdentityIQ stores the service request number for later audit and review.

Troubleshooting

This section provides the resolutions for the following errors that may be encountered while setting up and configuring ServiceNow Service integration Module.

1 - 'Authorization Required' error messages

The following type of error messages appearwhen the authorization data is not sent to ServiceNow:

```
Caused by: org.apache.axis2.AxisFault: Transport error: 401 Error:
Authorization Required
```

Resolution: Verify the procedure to configure appropriate authorization mechanism. For more information on the procedure, see "Configuration procedure" on page 93.

2 - Document Type Declaration (DTD) parsing errors

The DTD parsing errors appear when the following JVM arguments are not defined for your application server:

- **-Djavax.xml.soap.SOAPConnectionFactory=org.apache.axis2.saaj.SOAPConnectionFactoryImpl**
- **-Djavax.xml.soap.MessageFactory=org.apache.axis2.saaj.MessageFactoryImpl**
- **-Djavax.xml.soap.SOAPFactory=org.apache.axis2.saaj.SOAPFactoryImpl**

Resolution: Ensure that the application is pointing to the correct java runtime (that is, 1.6) and the above mentioned JVM arguments are defined for application server.

Note: Enter the following command to enable log4j tracing on this component:
log4j.logger.sailpoint.integration.servicenow =debug,file

3 - For certificate based authentication the IdentityIQ Server and ServiceNow instance must have correct time set.

For certificate based authentication, ensure that the IdentityIQ Server and ServiceNow instance have the correct date, time, and timezone set.

4 - When the Test Connection fails error messages are displayed in IdentityIQ and log file of WebSphere

The following error mesage is displayed in IdentityIQ:

```
Unable to engage module : rampart
```

The following error mesage is displayed in the log file of WebSphere:

```
ERROR WebContainer : apache.axis2.deployment.ModuleDeployer:113 - The
rampart-1.6.1.mar module, which is not valid, caused Could not initialize
class org.apache.axis2.deployment.util.TempFileManager
```

Resolution: If the above error message is diplayed in the log file of WebSphere, set the temporary directory in **Generic JVM arguments** of **Java Virtual Machine** by setting the following variable:

-Djava.io.tmpdir=<FullPathOfTempDir>

Note: **Ensure that the UNIX user where WebSphere is installed should be the owner of the temporary directory.**

Appendix A: Common Identity Management Integration Configuration

This appendix describes the following information.

Overview

This appendix describes configuration process for integrations with identity management (IDM) systems and the places in IdentityIQ that use the integrations. It does not describe the details of a specific integration only the general framework common to all integrations.

Creating the IntegrationConfig Object

The first step in configuring an integration is designing an instance of the IntegrationConfig object. There is currently no user interface for editing these objects, you must write them in XML and import them. The IntegrationConfig defines the following things:

- Java class that handles communication with the IDM system
- Connection parameters such as host name, user name, and password
- IdentityIQ Application object that represents the IDM system in aggregations
- List of the applications that are managed by the IDM system
- Resource and attribute name mappings
- Role synchronization style
- Methods for selecting roles to synchronize

Here is an example of **IntegrationConfig** file that has all of the options:

```
<IntegrationConfig name='Example Integration'
  executor='sailpoint.integration.ExampleIntegration'
  roleSyncStyle='it'>

<!--
 Application representing the IDM system in IIQ
-->
<ApplicationRef>
 <Reference class='Application' name='Example Integration'/>
</ApplicationRef>

<!--
 Connection parameters needed by the executor.
-->

<Attributes>
 <Map>
  <entry key='url' value='http://somehost:8080/rest/iiq'/>
```

```
    <entry key='username' value='jlarson'/>
    <entry key='password' value='1:987zxd9872970293874'/>
  </Map>
</Attributes>

<!--
  Definitions of managed resources and name mappings.
-->
<ManagedResources>
 <ManagedResource name='LDAP 42'>
  <ApplicationRef>
   <Reference class='Application' name='Corporate Directory'/>
  </ApplicationRef>
  <ResourceAttributes>
   <ResourceAttribute name='memberOf' localName='groups'/>
  </ResourceAttributes>
 </ManagedResource>
</ManagedResources>

<!--
  Synchronized role list.
  In practice you will never have a SynchronizedRoles with
  the RoleSyncFilter or RoleSyncContainer elements. All three
  are shown only as an example.
-->
<SynchronizedRoles>
  <Reference class='Bundle' name='role1'/>
  <Reference class='Bundle' name='role2'/>
</SynchronizedRoles>
```

The executor attribute has the name of a class that implements the sailpoint.object.IntegrationExecutor interface. This class is conceptually similar to a Connector class in that it does the work specific to a particular integration. Each integration package will come with an example IntegrationConfig that contains the executor class name.

The roleSyncStyle attribute defines how roles are synchronized between IdentityIQ and the IDM system. The possible values are:

- **none**: roles are not synchronized
- **detectable**: detectable (IT) roles are synchronized
- **assignable**: assignable (business) roles are synchronized
- **dual**: both detectable and assignable roles are synchronized

If this attribute has no value the default is none. More information on the role synchronization process is found in the Role Synchronization section.

ApplicationRef

Some integrations support identity aggregation. In these cases there is a **sailpoint.object.Application** object defined to represent the IDM system and an implementation of the **sailpoint.connector.Connector** interface that handles communication with the IDM system. This is normally a multiplexed connector that returns objects representing the IDM system account as well as accounts on managed resources. Links in the identity cube are created for the managed resource accounts as well as the IDM system account.

```
<ApplicationRef>
  <Reference class='Application' name='Example Integration'/>
</ApplicationRef>
```

The documentation of each integration must describe the supported configuration attributes.

The following attributes are reserved and can only be used for the purposes defined here.

- **roleSyncHistory**: list of objects containing a history of previous synchronizations
- **universalManager**: enables the integration as a manager of all applications

The **roleSyncHistory** attribute contains a list of **sailpoint.object.IntegrationConfig.RoleSyncHistory** objects that have information about roles previously synchronized with this integration. This includes the name of the role and the date it was synchronized. This list can be used by the role synchronization task to optimize communication with the IDM system by sending only the roles that have changed since the last synchronization.

The **universalManager** attribute is set to the string true to enable this integration as a manager for all IdentityIQ applications without a ManagedResources list. This is intended primarily for testing purposes.

ManagedResources

If the integration supports provisioning, it must define a list of managed resources that corresponding to applications defined in IdentityIQ. This determines how provisioning plans created during certification or role assignment are divided and sent to each integration.

```
<!--
   Definitions of managed resources and name mappings.
 -->
 <ManagedResources>
  <ManagedResource name='LDAP 42'>
   <ApplicationRef>
     <Reference class='Application' name='Corporate Directory'/>
   </ApplicationRef>
   <ResourceAttributes>
    <ResourceAttribute name='memberOf' localName='groups'/>
   </ResourceAttributes>
  </ManagedResource>
 </ManagedResource>
```

The ManagedResources element contains a list of ManagedResource elements. A ManagedResource element must contain an ApplicationRef that defines the associated IdentityIQ application. The ManagedResource element might have an optional name attribute that defines the name of the resource within the IDM system. If the name is not specified it is assumed that the resource name is the same as the IdentityIQ application name.

The ManagedResource element might also contain a ResourceAttributes element that contains one or more ResourceAttribute elements. ResourcAttribute is used to define mappings between attribute names in the IDM system and IdentityIQ. ResourceAttribute has the following XML attributes.

- **name**: attribute name in the IDM system
- **localName**: attribute name in the IdentityIQ application schema

If a provisioning plan is sent to this integration with attributes that are not in the ResourceAttributes list it is assumed that the name in IdentityIQ is the same as the name in the IDM system.

The ResourceAttributes list does not define a filter for attributes sent to the IDM system it only defines name mappings. When an integration has an IdentityIQ application in the ManagedResource list it is assumed that all attribute requests for that application are sent to that integration. You cannot have more than one integration managing different sets of attributes for the same application.

There is a special attribute that can be defined in Attributes that declares the integration as the manager of all applications in IdentityIQ regardless of the content of the ManagedResources element. You can still use ManagedResources to define name mappings for certain applications when necessary.

SynchronizedRoles

The SynchronizedRoles element is used to define a concrete list of roles to consider when roles are synchronized. When this is included in an IntegrationConfig object it has priority over both RoleSyncFiler and RoleSyncContainer elements if they are also included.

```
<SynchronizedRoles>
  <Reference class='Bundle' name='role1'/>
  <Reference class='Bundle' name='role2'/>
</SynchronizedRoles>
```

This can be used to synchronize simple integrations with a small set of roles that does not change often. If the set of roles is large or frequently changing, it is better to use RoleSyncFilter or RoleSyncContainer.

RoleSyncFilter

The RoleSyncFilter element contains a filter that is used to identify the roles to consider for synchronization.

```
<RoleSyncFilter>
  <Filter property='syncFlag' operation='EQ' value='true'/>
</RoleSyncFilter>

<RoleSyncFilter>
  <Filter property='name' operation='LIKE' matchMode='START'
value='Sync'/>
</RoleSyncFilter>
```

Synchronization filters typically used extended role attributes. In the first example an extended attribute syncFlag must be configured and have a value of true for the role to be synchronized.

Naming conventions can also be used to identify synchronized roles. In the second example any role whose name starts with Sync is synchronized.

A RoleSyncFilter can be combined with a RoleSyncContainer. If both are specified the intersection of the two role sets is considered for synchronization.

RoleSyncContainer

The RoleSyncContainer element defines the set of roles to be considered for synchronization by identifying an inherited role.

In this example, any role that directly or indirectly inherits the role named Roles To Synchronize is synchronized. This is typically a container role that has no function other than organizing other roles.

Specifying roles with inheritance has the advantage of creating a node in the modeler tree for Roles To Synchronize that you can expand to quickly see all of the synchronized roles.

A RoleSyncContainer can be combined with a RoleSyncFilter. If both are specified the intersection of the two role sets are synchronized.

Note: **If an IntegrationConfig does not have any SynchronizedRoles, RoleSyncFilter, or RoleSyncContainer elements and the roleSyncStyle element has a value other than none, it is assumed that all roles are considered for synchronization.**

Aggregation

Some integrations support feeds of identity information through the normal aggregation process. In these cases the integration package will have a IdentityIQ .connector.Connector implementation class and an example IdentityIQ .object.Application object in XML.

IDM connectors are usually multiplexed connectors that return objects representing the IDM system account as well as accounts on all managed resources.

When an aggregation application is defined a reference to it should be placed in the IntegrationConfig. This enables provisioning operations to obtain the account name in the IDM system that corresponds to an identity in IdentityIQ.

Role Synchronization

Role synchronization is performed by running the standard system task named Synchronize Roles. This task is defined in the file **tasksRunnable.xml** and is created during the normal initialization and upgrade processes.

The task normally attempts synchronization with every **IntegrationConfig** stored in the repository. The task has one hidden input argument, integrations, that can be set to a CSV of names of IntegrationConfig objects. Use this to restrict the synchronization to a particular set of integrations.

Each **IntegrationConfig** has a **roleSyncStyle** attribute that determines how roles are synchronized. If this attribute is missing or set to none, role synchronization is disabled.

When synchronization is enabled, the task first determines the set of candidate roles by evaluating the filtering options defined in the IntegrationConfig. If there is a SynchronizedRoles element it defines the concrete list of candidate roles. Otherwise the RoleSyncFilter and RoleSyncContainer are evaluated and intersected to produce the set of candidate roles.

Note: Candidate roles are not necessarily the ones that are sent to the IDM system. The roles sent are further constrained by the synchronization style.

roleSyncStyle=detectable

When the synchronization style is detectable the candidate role list is filtered to contain only detectable roles as defined by the role type.

For each candidate role a simplified representation of the role is built using the IdentityIQ **.integration.RoleDefinition** class, this is called the target role.

Entitlements for the target role are extracted from the candidate in one of two ways. If the candidate role has a provisioning plan, the plan defines the resources and attribute values that are included in the target role. If the candidate role has no provisioning plan, a set of resource attributes is derived by analyzing the profile filters in the candidate role.

To give a role a provisioning plan, design an instance of the IdentityIQ **.object.ProvisoningPlan** as an XML and place it inside the XML for the IdentityIQ **.object.Bundle** object representing the role.

Using provisioning plans gives you more control over the contents of the target role. Profile filters might be ambiguous or result in more attribute values for the role than are necessary, but for relatively simple filters it can be easier than defining provisioning plans.

To derive target roles, first build a list of candidate profiles. If the role option orProfiles is false, than all profiles defined in the role become candidates. If the orProfiles option is true, then only the first profile in the role is a candidate. Then iterate over each filter in each candidate profile applying this algorithm:

if the filter is EQ or CONTAINS_ALL

```
  add the values for this attribute comparison to the role
if the filter is OR
  recurs for the first child filter term
if the filter is AND
  recurs for all child filter terms
```

If the role inherits other roles, the hierarchy is flattened and inherited entitlements are merged into the target role. The same process described above is applied to every role in the inheritance hierarchy.

roleSyncStyle=assignable

When the synchronization style is assignable the candidate role list is filtered to contain only assignable roles as defined by the role type.

For each candidate role, build a simplified representation of the role using the IdentityIQ .integration.RoleDefinition class, this is called the target role.

Entitlements for the target role are extracted from the candidate by first applying the process described in roleSyncStyle=detectable to the candidate role. This might not have any effect since assignable roles do not normally have provisioning plans or profiles.

Next the roleSyncStyle=detectable process is applied to each of the required roles referenced by the candidate role. This is typically where most of the entitlements are found.

roleSyncStyle=dual

This is a hybrid of the assignable and detectable styles used only by the IBM Tivoli Identity Manager integration.

First detectable roles are synchronized as defined in the “roleSyncStyle=detectable” on page 101 section.

Next assignable roles are synchronized. Instead of entitlements the roles have an extended attribute containing the names of all roles that were on the required list. The integration executor might further annotate the role definition with rules for automated assignment.

Provisioning

Provisioning can be performed in several ways.

- After role assignment from the IdentityIQ identity edit page
- After role assignment from the Access Request Manager
- During certification to handle revocations and role completions
- In a background reconciliation task
- During aggregation

Both IdentityIQ and the Access Request Manager (ARM) launch workflows with provisioning being done at the end. This provides the opportunity to insert an approval step before provisioning. The default workflow for IdentityIQ identity edits is named Identity Update. By default it has no approvals but does attempt provisioning. The example workflow for ARM requests is named ARM Role Approval Example.

Certifications can do provisioning to remove entitlements and roles that were revoked as well as add missing entitlements that are necessary to satisfy a role assignment.

A reconciliation task is an instance of the Identity Refresh task template with the provisioning argument set to true. This argument is visible in the configuration page for the refresh task. Reconciliation compares the assigned roles with the detected entitlements and automatically provisioning any missing entitlements. Entitlements might be missing due to either changes in role assignments for an identity, or changes to the definition of roles already assigned to an identity.

Reconciliation is intended to replace the IdentityIQ Provisioning. The old provisioning page was role oriented, monitored changes to roles, and sent provisioning requests for users assigned to modified roles. It did not detect changes to the assigned roles list of identities, however. The reconciliation task is identity oriented and calculates all changes necessary to make an identity's entitlements match the currently assigned roles.

Since reconciliation is now part of the core set of identity refresh options, it can also be done during aggregation. This is less common, but aggregation could change account attributes that are used by role assignment rules resulting in changes to the assigned and detected role lists. With provisioning enabled, the aggregation could trigger the provisioning of missing entitlements for the assigned roles. A common use case for this would be aggregating from an application representing a HR system with HR attributes determining assigned business roles.

Note: Automated provisioning done by the reconciliation task or within workflows typically does not remove entitlements, it only adds missing entitlements. Removal of unnecessary entitlements is expected to be done in a certification where a user has more control. While it is possible to enable removals during automated provisioning, it is potentially dangerous and should not be done without careful consideration.

Provisioning with Synchronized Roles

The provisioning sub-system works with the role synchronization sub-system to determine how role assignments are provisioned. If roles are not being synchronized, the raw entitlements needed by that role are compiled and sent to the integration executors. If an integration supports role synchronization, requests are compiled to the IDM system itself to add or remove native role assignments.

There might be median cases where an integration does role synchronization, but only manages a subset of the possible applications. In these cases plans are compile with both IDM system role assignments and as raw entitlements for the entitlements that are not covered by a native role assignment.

Index